FastHenry USER'S GUIDE

Version 3.0

M. Kamon L.M. Silveira C. Smithhisler J. White

Research Laboratory of Electronics
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139 U.S.A.

11 November 1996

This work was supported by Defense Advanced Research Projects Agency contract N00014-91-J-1698, a National Science Foundation Graduate Fellowship, and grants from IBM and Digital Equipment Corporation.

Contents

This manual describes FastHenry, a three-dimensional inductance extraction program. FastHenry computes the frequency dependent self and mutual inductances and resistances between conductors of complex shape. The algorithm used in FastHenry is an acceleration of the mesh formulation approach. The linear system resulting from the mesh formulation is solved using a generalized minimal residual algorithm with a fast multipole algorithm to efficiently compute the iterates. See Appendix A for references.

This manual is divided into four sections. The first section explains the syntax for preparing input files for FastHenry. The input files contain the description of the conductor geometries. The second section describes how to run the program and process the output. The third section describes the generation of postscript images to visualize the three dimensional geometries defined in the input file. The fourth section shows how to use Matlab to observe current distribution in reference plances.

The files "nonuniform_manual_*.ps" constitute an important supplement to this manual and describe the new routines for specifying a nonuniformly discretized reference plane.

Information on compiling FastHenry, obtaining the FastHenry source code, producing this document, and corresponding about FastHenry is given in Appendix A.

Appendix B summarizes most of the changes in this version, 3.0, over the previous version, 2.5.

1 How to Prepare Input Files

This section of the manual describes how to prepare input files for FastHenry. The input files specify the discretization of conductor volumes into filaments. The input file specifies each conductor as a sequence of straight segments, or elements, connected together at nodes. Each segment has a finite conductivity and its shape is a cylinder of rectangular cross section of some width and height. A node is simply a point in 3-space. The cross section of each segment can then be broken into a number of parallel, thin filaments, each of which will be assumed to carry a uniform cross section of current along its length. The first two parts of this section describe the file format through simple examples. A detailed description is in the third part, and more complex examples can be found in the last section and later in the manual.

1.1 A Simple Example

The following is an input file which calculates the loop inductance of four segments nearly tracing the perimeter of a square. It is described more thoroughly on the next page.

```
**This is the title line. It will always be ignored**.
* Everything is case INsensitive
* An asterisk starts a comment line.

* The following line names millimeters as the length units for the rest
* of the file.
.Units MM
```

```
* Make z=0 the default z coordinate and copper the default conductivity.
* Note that the conductivity is in units 1/(mm*Ohms), not 1/(m*Ohms)
* since the default units are millimeters.
.Default z=0 sigma=5.8e4

* The nodes of a square (z=0 is the default)
N1 x=0 y=0
N2 x=1 y=0
N3 x=1 y=1
N4 x=0 y=1
N5 x=0 y=0.01

* The segments connecting the nodes
E1 N1 N2 w=0.2 h=0.1
E2 N2 N3 w=0.2 h=0.1
E3 N3 N4 w=0.2 h=0.1
E4 N4 N5 w=0.2 h=0.1

* define one 'port' of the network
.external N1 N5

* Frequency range of interest.
.freq fmin=1e4 fmax=1e8 ndec=1

* All input files must end with:
.end
```

As described in the comments, `.Units MM` defines all coordinates and lengths to be in millimeters. All lines with an `N` in the first column define nodes, and all lines starting with `E` define segments. In particular, the line

```
E1 N1 N2 w=0.2 h=0.1
```

defines segment E1 to extend from node `N1` to `N2` and have a width of 0.2 mm and height of 0.1 mm as drawn in Figure 1. If the $n \times n$ impedance matrix, $Z(\omega)$, for an n-conductor problem is thought of as the parameters describing an n-port network, then the line

```
.external N1 N5
```

defines `N1` and `N5` as one port of the network. In this example, only one port is specified, so the output will be a 1×1 matrix containing the value of the impedance looking into this one port.

FastHenry calculates $Z(\omega)$ at the discrete frequencies described by the line

```
.freq fmin=1e4 fmax=1e8 ndec=1
```

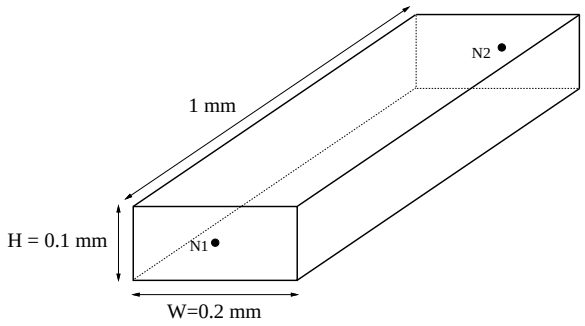

Figure 1: Example Segment for Sample Input File

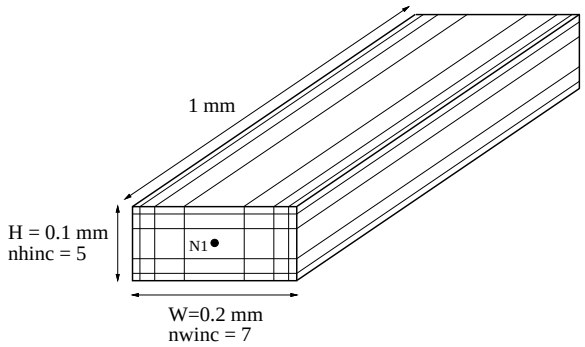

Figure 2: Segment discretized into 35 filaments

where `fmin` and `fmax` are the minimum and maximum frequencies of interest, and `ndec` is the number of desired frequency points per decade. In this case, $Z(\omega)$ will be calculated at $10^4, 10^5, 10^6, 10^7$, and 10^8 Hz. All input files must end with `.end`.

In the above example, FastHenry created one filament per segment since no discretization of the segments into filaments was specified. In order to properly model non-uniform cross sectional current due to skin and proximity effects, a finer discretization must be used. Finer filaments are easily specified in the segment definition. For example, replacing the definition for `E1` with

```
E1 N1 N2 w=0.2 h=0.1 nhinc=5 nwinc=7
```

specifies that `E1` is to be broken up into thirty-five filaments: five along its height (`nhinc=5`) and seven along its width (`nwinc=7`). See Figure 2.

1.2 Another Simple Example

To continue teaching by example, what follows is an example of computing the loop inductance of an L shaped trace over a ground plane with the trace's return path through the plane as shown in Figure 3. Note that a line beginning with '+' is a continuation of the previous line.

```
* A FastHenry example using a reference plane
```

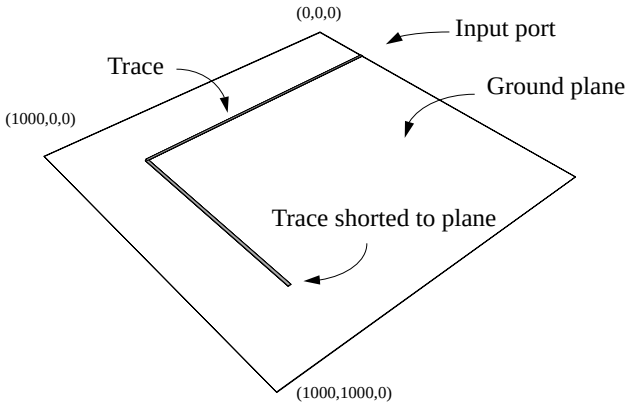

Figure 3: L-shaped trace over ground plane

```
*  This example defines an L shaped trace above a ground plane which
*  has a return path through the plane

*  Set the units for all following dimensions
.units mils
*
* Define ground plane and nodes to reference later
*
*  First define 3 corners of the plane, in clockwise or counter-cw order
g1 x1 = 0     y1 = 0     z1 = 0
+  x2 = 1000  y2 = 0     z2 = 0
+  x3 = 1000  y3 = 1000  z3 = 0
*     thickness:
+  thick = 1.2
*     discretization:
+  seg1 = 20 seg2 = 20
*     nodes to reference later:
+  nin  (800,800,0)
+  nout (0,200,0)

* Some defaults to model skin effects
.default nwinc=8 nhinc=1
*
* L shaped trace over ground plane
*
* The nodes
N1 x=0 y=200 z=1.5
N2 x=800 y=200 z=1.5
N3 x=800 y=800 z=1.5

* The elements connecting the nodes
```

```
E1 N1 N2 w=8 h=1
E2 N2 N3 w=8 h=1

* Short together the end of the L shaped trace (N3) and its corresponding
*  point on the ground plane directly beneath (nin)
.equiv nin n3

* compute loop inductance from beginning of L (N1) to its corresponding
*   point directly underneath (nout)
.external N1 nout

* Compute impedance matrix for one very low frequency (essentially DC)
*   and one very high frequency
.freq fmin=1e-1 fmax=1e19 ndec=0.05

* mark end of file
.end
```

In this example, the ground plane is a $1000mil \times 1000mil$ sheet of copper (by default) defined by three of its four corners, $(0,0,0)$, $(1000,0,0)$, $(1000,1000,0)$. The plane is 1.2 mils thick and its discretization is specified by `seg1` and `seg2` (see Section 1.3.9). The discretization of the plane forms a grid of nodes and interconnecting segments. `nin` and `nout` refer to the internal nodes of the plane which are closest to $(800,800,0)$ and $(0,200,0)$, respectively.

To model skin effects on the trace, the default for the number of filaments per segment is set to 8.

To compute loop inductance, the node at one end of the trace is shorted to the plane by declaring the the node to be "electrically equivalent" to the node directly underneath.

```
.equiv nin n3
```

and then the other end is declared as the "port" with the `.external` statement.

In this case, a single loop impedance will be computed. If, however, the `.equiv` and `.external` were replaced with

```
.external N1 N3
.external nout nin
```

then the partial inductances and resistances of these two paths would be computed yielding a 2×2 impedance matrix.

Computing the impedance for only two frequencies is useful for visualization of the distribution of current in the reference plane as described later in Section 4.

1.3 Input File Syntax

The previous section described many of the basics required for an input file. This section gives a more complete and detailed description of the input file format and should serve as a reference.

Some general facts about file syntax:

- Lines are processed sequentially.
- Uppercase is converted to lower case.
- "*" at the beginning of a line marks a comment line.
- Lines are restricted to 1000 characters but can be continued with a "+" as the first character of subsequent lines. Intervening "*" lines are allowed in this release.
- The first line in the file is considered the title line and is ignored. It is recommended that this line start with an "*" for future compatibility and file concatenation.
- The file must end with the `.End` keyword.
- Names of objects are limited to 80 characters.

In general, each line of the input file will either define some geometrical object, such as a node or segment, or it will specify some program parameter. All input lines that define geometrical objects begin with a letter defining their type, and then some unique alphanumeric string of up to 80 characters. For instance, all node definitions begin with the letter `N`. This sets object lines apart from parameter specification lines which begin with a period, ".".

The remainder of this section will describe all possible input lines. In the following description, any argument enclosed in '[ ]' indicates an optional argument. If not included on the input line, the actual value used for this argument will be either the program default, or the user default defined by the `.Default` keyword (described below).

1.3.1 Node Definitions

```
Syntax:  Nstr [x = x_val] [y = y_val] [z = z_val]
```

This defines a node called `Nstr` where `str` is any alphanumeric string. The first character on the line must begin with an `N` for this to be interpreted as a node definition. The node will have location (x_val, y_val, z_val) where each coordinate has units defined by the `.Units` keyword.

Any of the coordinates can be omitted assuming that a default value has been previously specified with the `.Default` keyword. Otherwise, an error will occur and the program will exit.

1.3.2 Segment Definitions

```
Syntax:  Estr node1 node2 [w = value] [h = value] [sigma, rho = value]
                 [wx = value wy = value wz = value]
                 [nhinc = value] [nwinc = value] [rh = value] [rw = value]
```

This defines a segment called `Estr` where `str` is any alphanumeric string. The first character on the line must begin with the letter `E` for this to be interpreted as a segment definition. The segment will extend from node `node1` to node `node2` which must be previously defined node names. `h` and `w` are the segment height and width. Either `sigma`, the conductivity, or `rho`, the resistivity, can be specified for the segment.

Discretization of the segment into multiple, parallel thin filaments is specified with the `nhinc` and `nwinc` arguments. `nhinc` specifies the number of filaments in the height direction, and `nwinc`, the number in the width direction. Both must be integers. See Figure 2. By default, the ratio of adjacent filaments is 2.0 as in Figure 2, however this can changed with the `rh` and `rw` arguments which specify the ratio in the height and width direction, respectively. While an automatic technique of dividing a section into filaments based on the skin depth may be a better approach, no such feature is available. See Section 2.4 for guidance in choosing filament discretizations.

To specify the orientation of the cross section, `wx`, `wy`, and `wz` represent any vector pointing along the width of the segment's cross section. If these are omitted, the width vector is assumed to lie in x-y plane perpendicular to the length. If the length direction is parallel to the z-axis, then the width is assumed along the x-axis.

`h` and `w` can be omitted provided they are assigned a default value in a previous `.Default` line.

`nhinc`, `nwinc`, and `sigma` or `rho` can be omitted, and if not previously given a default value, then 1, 1, and the conductivity of copper, respectively, are used as default values.

Note that the nodes used to define the elements must be defined under **Node Definitions** described above and cannot be reference plane nodes (See Section 1.3.9 for a description of reference planes). To connect to a reference plane, the user must instead create a new node at the desired location as described in **Node Definitions** and then use the `.Equiv` keyword to equivalence the reference plane node and the new node. For instance, the following is not allowed:

```
g1 x1 = 0     y1 = 0     z1 = 0
+  x2 = 1000  y2 = 0     z2 = 0
  .
  .
+ n_gp (5,5,0)

N1 x=5 y=5 z=10
E1 N1 n_gp
```

The legal way to define the segment would be

```
g1 x1 = 0     y1 = 0     z1 = 0
+  x2 = 1000  y2 = 0     z2 = 0
```

```
  .
  .
+ n_gp (5,5,0)

N1 x=5 y=5 z=10
N2 x=5 y=5 z=0
E1 N1 N2
.equiv N2 n_gp
```

1.3.3 `.Units` keyword

```
Syntax:  .Units unit-name
```

This specifies the units to be used for all subsequent coordinates and lengths until the end of file or another `.Units` specification is encountered. Allowed units are kilometers, meters, centimeters, millimeters, micrometers, inches, and mils with `unit-name` specified as `km, m, cm, mm, um, in, mils`, respectively.

Note that this keyword affects the expected units for the conductivity and resistivity.

1.3.4 `.Default` keyword

```
Syntax:  .Default [x = value] [y = value] [z = value] [w = value]
                  [h = value] [sigma, rho = value]
                  [nhinc = value] [nwinc = value] [rh = value] [rw = value]
```

This keyword specifies default values to be used for subsequent object definitions. A certain default value is used until the end of the file, or until it is superseded by another `.Default` line changing that value.

1.3.5 `.External` keyword

```
Syntax:  .External node1 node2 [portname]
```

This keyword specifies node `node1` and node `node2` as a terminal pair or port whose impedance parameters should be calculated for the output impedance matrix. If an input file includes n `.External` lines, then the impedance matrix will be an $n \times n$ complex matrix. The port can be given a name by specifying a single word string `portname`. This name is used to reference this port in the output file, `Zc.mat`, and is also used with the command line `-x` option described in Section 2.3.

This keyword effectively places a voltage source between these nodes and will later use the current through that source to determine an entry in the admittance matrix. The first node specified is the positive node. Note that it is up to the user to insure that there are NO loops of only voltage sources. Also, a voltage source with no possible return path will always carry zero current producing a row of zeros in the admittance matrix. The output impedance matrix will thus be nonsense (NaN).

1.3.6 .Freq keyword

```
Syntax:  .Freq fmin=value fmax=value [ndec = value]
```

This keyword specifies the frequency range of interest. `fmin` and `fmax` are the minimum and maximum frequencies of interest, and `ndec` is the number of desired frequency points per decade. FastHenry must perform the entire solution process for each frequency.

Note that `ndec` need not be an integer. For instance,

```
.freq fmin=1e3 fmax=1e7 ndec=0.5
```

will have FastHenry calculate impedance matrices for $f = 10^3, 10^5$, and 10^7 Hz.

If `fmin` is zero, FastHenry will run only the DC case regardless of the value of `fmax`.

1.3.7 .Equiv keyword

```
Syntax:  .Equiv node1 node2 node3 node4 ...
```

This keyword specifies that nodes `node1, node2, node3, node4,...` are to be considered electrically equivalent yet maintain their separate spatial coordinates. The nodes are effectively 'shorted' together. If any of the node names are not previously defined, then they become pseudonyms for those nodes in the list which are defined. Note that the current flow between equivalenced nodes is not electromagnetically modeled.

1.3.8 .End keyword

```
Syntax:  .End
```

This keyword specifies the end of the file. All subsequent lines are ignored. This line must end the file.

1.3.9 Reference Plane definitions

Note: It is recommended that nonuniformly discretized planes be used instead of what is described here if possible. See the document "nonuniform_manual_*.ps". Uniform planes described below can still be used if they satisfy your needs. Also, for new users, understanding uniform planes is necessary to understand the nonuniform_manual.ps document.

Uniformly discretized planes

```
Syntax:  Gstr x1=value y1=value z1=value x2=value y2=value z2=value
              x3=value y3=value z3=value
              thick=value seg1=value seg2=value
              [segwid1 = value] [segwid2 = value]
              [sigma, rho = value]
              [nhinc=value] [rh=value]
              [relx=value] [rely=value] [relz=value]
              [Nstr1     (x_val,y_val,z_val) ]
```

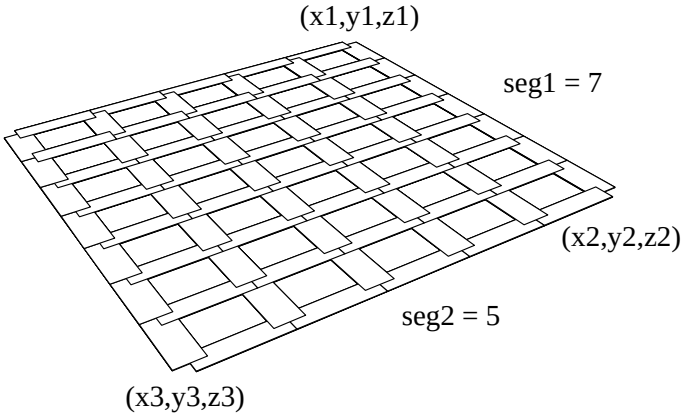

Figure 4: Discretization of a Reference Plane. Segments are one-third actual width.

```
[Nstr2     (x_val,y_val,z_val) ]
[Nstr3     (x_val,y_val,z_val) ].....
[hole <hole-type> (val1,val2,....)]
[hole <hole-type> (val1,val2,....)].....
```

This defines a uniformly discretized reference plane of finite extent and conductivity called `Gstr` where `str` is any alphanumeric string. The first character on the line must begin with the letter `G` for this to be interpreted as a reference (originally "ground") plane definition. The three locations $(x1, y1, z1), (x2, y2, z2), (x3, y3, z3)$ mark three of the four corners of the plane in either clockwise or counterclockwise order. FastHenry will determine the fourth corner assuming the first three are corners of a rectangle. The thickness of the plane is specified with the `thick` argument and the conductivity with either the `sigma` or `rho` argument. Note that current is modeled to flow two dimensionally, i.e., not in the thickness direction.

The reference plane is approximated by first laying down a grid of nodes, and then with segments, connecting every node to its adjacent nodes excluding diagonally adjacent nodes. Each segment is given a height equal to the specified thickness of the plane and, by default, width equal to the node spacing. This choice of width completely fills the space between segments. Figure 4 shows a sample reference plane with segments that are one-third full width for illustration.

`seg1` and `seg2` specify the number of segments along each edge of the plane. `seg1` is the number of segments along the edge from $(x1, y1, z1)$ to $(x2, y2, z2)$ and `seg2`, the number along the edge from $(x2, y2, z2)$ to $(x3, y3, z3)$. Thus the total number of nodes created will be (`seg1`+1)*(`seg2`+1), and the total number of segments created will be $(\texttt{seg1} + 1) * \texttt{seg2} + \texttt{seg1} * (\texttt{seg2} + 1)$. Note that the line from the node at $(x1, y1, z1)$ to the node at $(x2, y2, z2)$ define the edge of the plane, however segments that run from $(x1, y1, z1)$ to $(x2, y2, z2)$ will overhang the edge by half their width since nodes are defined at the center of segments. This slight overapproximation to the width of the plane is avoided in the nonuniform discretization code described in `nonuniform_manual_*.ps`.

`nhinc` and `rh` can be used to specify a number of filaments for discretization of each segment along the thickness (See Section 1.3.2). This could be used for modeling

nonuniform current across the thickness. If `nhinc` is omitted, the value of 1 is used regardless of the `.Default` setting. If `rh` is omitted, the default value is used.

Note: If you do not want to use segwid1 and segwid2 or holes as described below, you can use the nonuniform plane definition as described in "nonuniform_manual.ps".

Connections to the plane

Since the reference plane nodes are generated internally, there is no way to refer to them later in the input file. The exceptions to this are the nodes explicitly referenced in the reference plane definition. The argument

```
Nstr1     (x_val,y_val,z_val)
```

where `str1` is an alphanumeric string, will cause all subsequent references to `Nstr1` to refer to the node in the plane closest to the point (x_val, y_val, z_val). Note that *no spaces* are allowed between the '()'. The referencing is accomplished internally by effectively doing

```
.Equiv Nstr1 <internal-node-name>
```

where `<internal-node-name>` is the internal node name of the nearest reference plane node.

If one or more of `relx`, `rely`, and `relz` are specified, then the above node referencing instead chooses the node closest to $(x\_val + relx, y\_val + rely, z\_val + relz)$. In other words, `relx`, `rely`, and `relz` default to 0 if not specified.

A coarsely discretized plane (small `seg1`, `seg2`) may cause two different node references to refer to the same reference plane node. FastHenry will warn of such an event, but it is not an error condition.

Note: It is recommended that connections to the plane be done with the "contact equiv_rect" utility for nonuniformly discretized planes as described in "nonuniform_manual.ps".

Meshed Planes

By default, the width of the segments of the plane are chosen to fill the space between adjacent segments. However, to model meshed planes the width of the segments can be chosen to be smaller with the `segwid1` and `segwid2` arguments. This gives the plane the appearance of a "mesh" as shown is Figure 4. `segwid1` specifies the width of the segments that are in the direction parallel to the plane edge from $(x1, y1, z1)$ to $(x2, y2, z2)$. `segwid2` specifies the width along the edge from $(x2, y2, z2)$ to $(x3, y3, z3)$.

Note that if the section of conductor between meshes is large compared to the size of the mesh holes, this method of discretization may be too coarse. For instance, assume the mesh consists of only nine holes on a 3x3 grid. In such a situation, consider using the `hole` functions described below to make the specific holes on a plane with `seg1,seg2` $>>$ 3. See also Section 1.4.3.

Holes in the plane

Holes can be specified in the plane with

```
hole <hole-type> (val1,val2,val3,...)
```

where `<hole-type>` is the hole type and the `valn`'s make the list of arguments to be sent to the hole generating function. Holes are generated by first removing reference plane nodes, and then removing all segments connected to those nodes. The following describes the available hole generating functions:

```
hole point (x,y,z)
```

removes the node nearest to the point (x,y,z).

```
hole rect (x1,y1,z1,x2,y2,z2)
```

removes a rectangular region whose opposite corners are the nodes in the plane nearest (x1,y1,z1) and (x2,y2,z2).

```
hole circle (x,y,z,r)
```

removes the nodes contained within the circle of radius `r` centered at (x,y,z).

```
hole user1 (val1,val2,...)
```

calls the user defined function hole_user1() to remove nodes. user1 - user7 are available. The user can add the functions to the source file `hole.c` contained in the release. See the functions hole_rect(), hole_point(), and hole_circle() to see examples of the format for writing user hole functions.

Any shaped hole can be formed with a combination of hole directives. A few exceptions exist, however. Forming a hole that isolates or nearly isolates a section of the plane is not allowed. FastHenry warns of this with:

```
Warning: Multiple boundaries found around one hole region
  possibly due to an isolated or nearly isolated region of conductor.
  This may lead to no unique solution.
```

If a nearly isolated section of conductor is truly desired, consider defining two separate planes and connecting them.

Holes may not be produced as expected if the discretization of the plane is coarse. It is recommended that the plane be viewed by using the options "`-f simple -g on`" options to generate a `zbuf` file which can be used to generate a postscript image of the plane as described in Section 3.

1.4 Other Examples

This section describes other examples using more of the features of FastHenry. Some examples were created by the authors and some contributed from users. All files are available with the FastHenry source code.

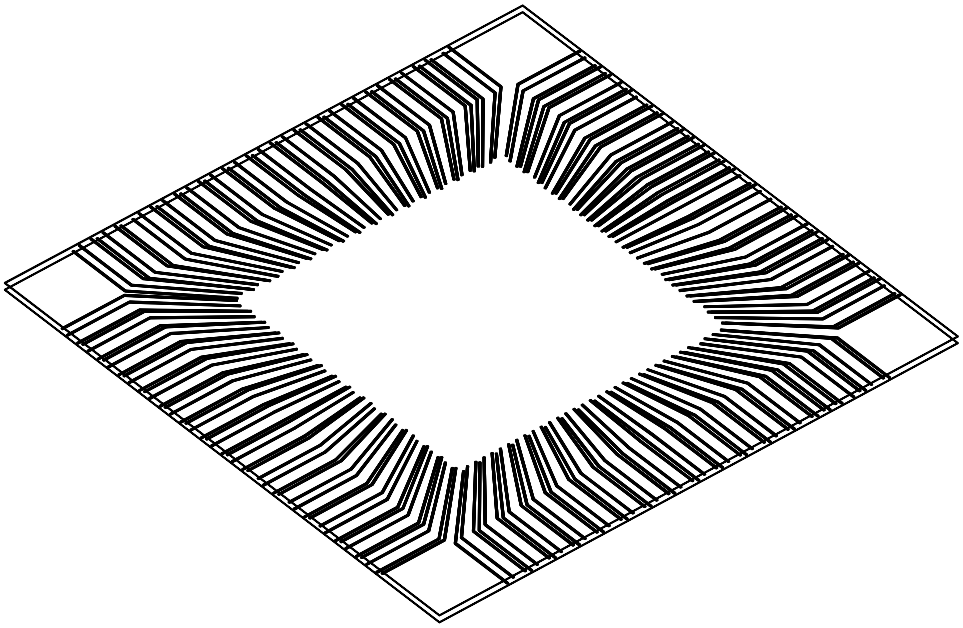

Figure 5: Printed Circuit Board example

1.4.1 Printed Circuit Board

The example file `gpexamp_copper.inp` shown in Figure 5 is a portion of a printed circuit board from Digital Equipment Corporation. This structure is to be placed underneath a PGA package. This geometry includes 2 reference planes sandwiching a large number of copper lines which lead to the center where the package would be placed. The set of copper lines are grouped into 18 groups according to functionality (see the many .equiv statements at the bottom of the file).

1.4.2 Vias and Meshed Planes

The example file `vias.inp` shown in Figure 6 shows three vias passing through a meshed ground plane. To form the mesh structure, the standard reference plane definition is used with the `segwid1` and `segwid2` arguments. Some of the dimensions for this example were taken from B.J. Rubin, "An electromagnetic approach for modeling high-performance computer packages," *IBM J. Res. Dev.*, Vol. 34, No. 4, July 1990.

1.4.3 Holey ground plane

The example file `holey_gp.inp` shown in Figure 7 is a punctured version of the reference plane example of section 1.2. The holes of the plane are formed with the `rect` and `circle` hole directives.

1.4.4 Pin-connect

The example file `pin-connect.inp` shown in Figure 8 is thirty-five pins of a 68-pin cerquad pin package from Digital Equipment Corporation.

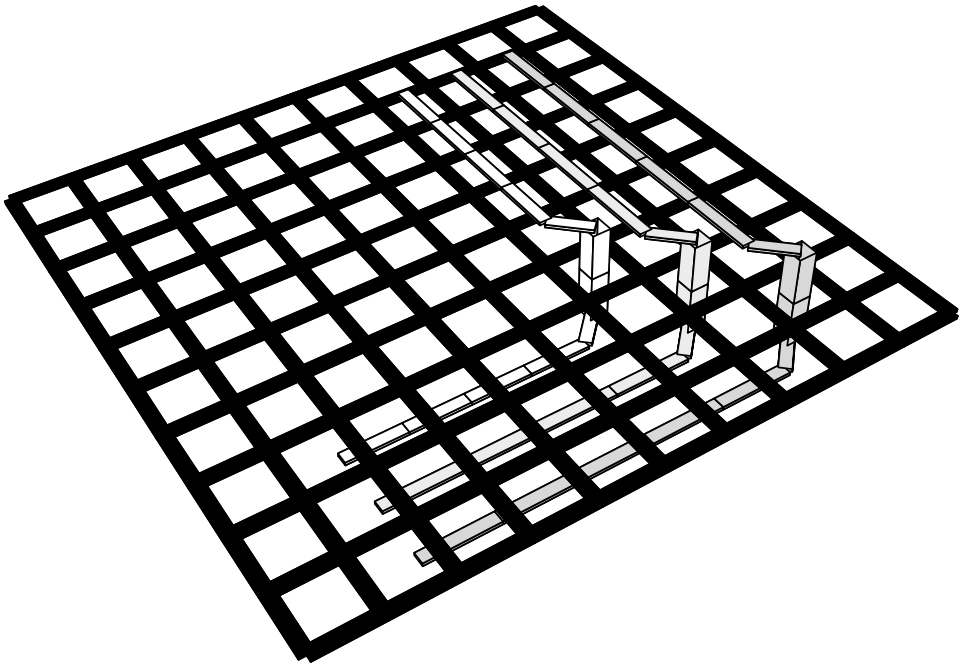

Figure 6: Vias through a meshed ground plane

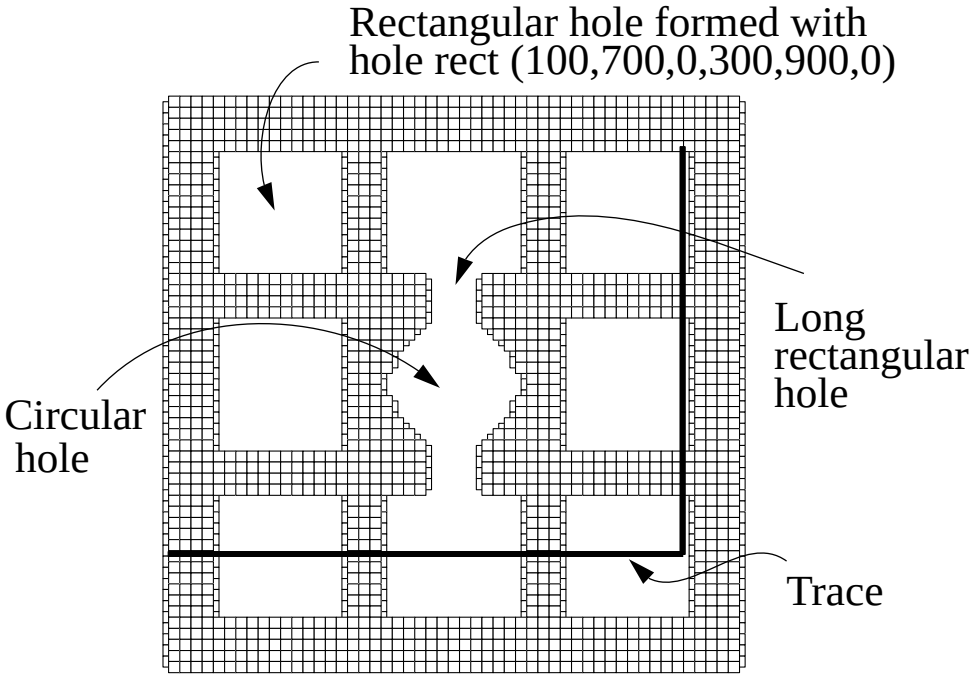

Figure 7: Trace over Ground Plane with holes