EMX User's Manual

Version 5.2
Integrand Software, Inc.
Copyright © 2004–2016

1 Introduction

EMX[*] is an electromagnetic simulator that is intended mostly for IC-style layouts. It reads an input GDSII file and computes Y-parameters. The simplest possible way of invoking EMX is like this:

```
emx layout.gds mystructure foundry.proc 1e9
```

This tells EMX to read the GDSII file `layout.gds` and to simulate the structure called `mystructure` within that file. Since the GDSII file does not contain information about the dielectric properties, this information must be supplied separately in a process file, which is called `foundry.proc` in this example. Finally, EMX needs the frequency at which to do the analysis (`1e9`, or 1 GHz). When EMX finishes, it prints the Y-parameters for the layout at that frequency.

2 Assumptions made by EMX

The conductors by default are assumed to have a thickness which is small relative to their widths, though EMX can also treat conductors as fully 3D. The conductors sit in a layered dielectric medium. Each layer has its own dielectric constant, conductivity, and height. All the layers are assumed to be infinite in the x and y directions. Beneath the bottom layer is an infinitely conductive ground plane, and the top layer extends to infinity in the z direction. The different metal layers may have different sheet resistances. Metals on different layers may be connected by vias which also have a finite conductivity.

EMX does not currently model the following:

1. Non-planar dielectrics.
2. Conductors that have a non-rectilinear cross section.
3. Conductors that connect by abutment, rather than through vias.

[*]EMX is a registered trademark of Integrand Software, Inc.

3 How EMX works

EMX uses the method of moments to extract the electrical behavior of the layout. The main steps in the method of moments are:

discretize The structure is split into small pieces (a “mesh”). Within each piece, which is called an element, the current and charge densities are assumed to have a simple form (a linear or constant function, for example).

compute interactions A bit of flowing current in an element creates a vector potential that varies throughout space. Similarly, a bit of charge in an element creates a scalar potential. The potentials are given by “Green’s functions” which are functions of the source and observation positions. EMX computes the vector and scalar potentials at the position of each element by summing up the contributions of the currents and charges of all the other elements. The potentials are arranged in square matrices whose size is equal to the number of elements.

solve linear system Together, the potentials determine the electric field. The electric field causes currents to flow according to Ohm’s law. Charge and current are related by charge conservation. Combining all of these relations yields a system of linear equations relating the current to the voltage source values. EMX solves this linear system to produce the electrical parameters.

The layout description for EMX is standard GDSII files. The mapping from GDSII layers to physical metal layers is given in the process file, which is described in the next section. Ports in the layout are indicated by labels in the GDSII file. EMX’s ports are described in section 6.

4 The process file

An example process file is shown in figure 1, and the corresponding cross section is shown in figure 2. The cross section does not show the full extent of the bottom or top layers. There is a ground plane beneath the bottom layer, and the top layer is air extending to infinity.

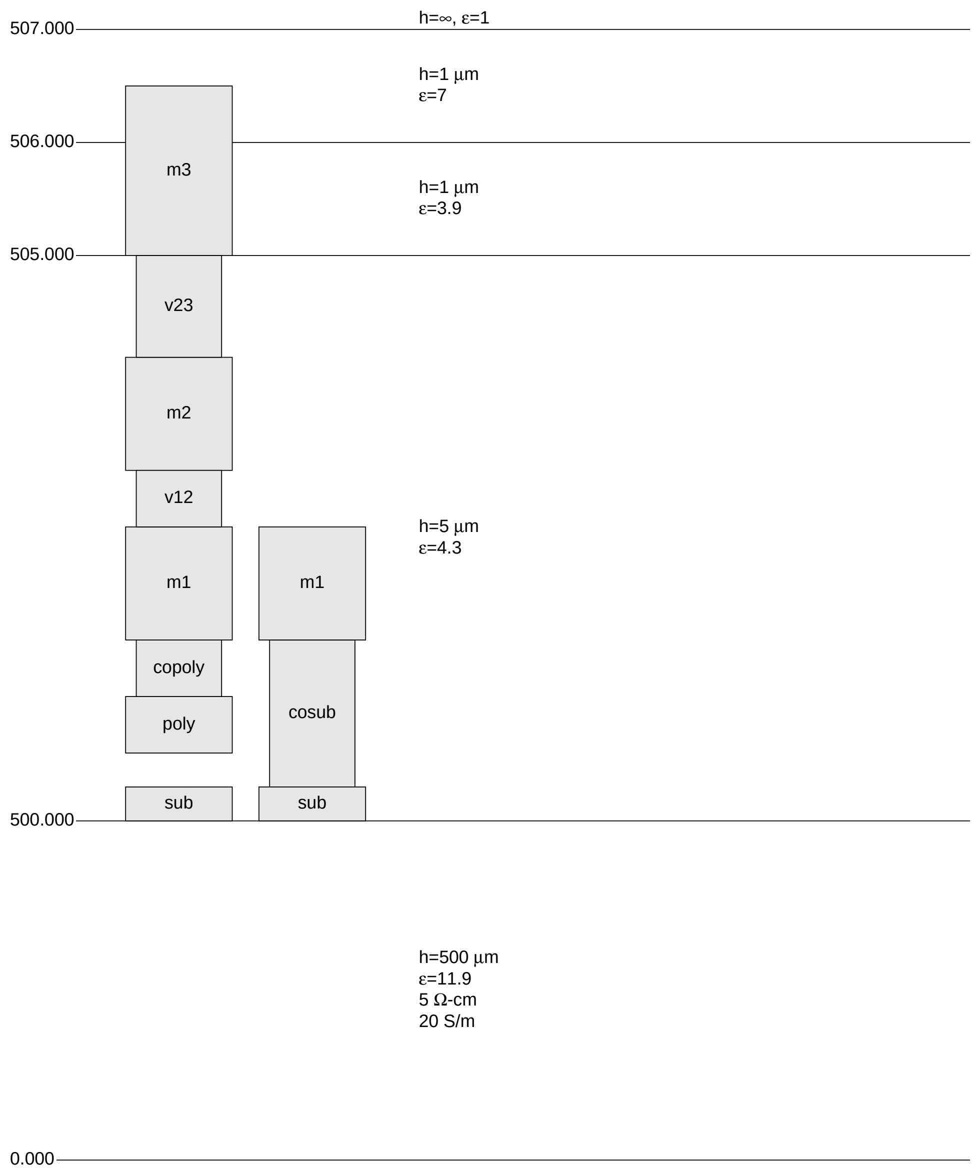
4.1 Layers, conductors, and vias

Each layer is indicated by the keyword `layer`, which is followed by up to four numbers: the height of the layer, the dielectric constant and magnetic permeability, and the conductivity of the layer. The height of the layer and the dielectric constant are required. If the magnetic permeability is omitted, it defaults to one, and if the conductivity is omitted, it defaults to zero. Dielectrics may also have a fixed loss tangent, specified as follows:

```
layer   10   3.9   tan delta 1.2e-4
```

```
# Comments start with '#' and continue to
# the end of the line

assume microns   # lengths will be in microns

# These are the GDSII layers for the conductors

define sub = L45T0
define poly = L47T0
define m1 = L49T1
define m2 = L51T1
define m3 = L53T1
define cosub = L48T1-poly
define copoly = L48T1*poly
define v12 = L50T1
define v23 = L52T1

# Dielectrics and conductors

layer             500     11.9   1        5 ohm-cm
layer               5      4.3
        conductor 0.3  50 ohm/sq  sub
        offset    0.6
        conductor 0.5  3.5e5   poly
        offset      1
        conductor   1  2.5e7   m1
        offset    1.5
        conductor   1  2.5e7   m2
layer               1      3.9
        conductor 1.5  2.5e7   m3
layer               1      7.0
layer        infinity      1.0

# L48T1 contacts sub except where blocked by poly
via     sub  m1   2.5e5   cosub
via     poly m1   2.5e5   copoly
via     m1   m2   2.5e5   v12
via     m2   m3   2.5e5   v23
```

Figure 1: Example process file



m3: h=1.5 µm, 26.7 mΩ/sq
m2: h=1 µm, 40 mΩ/sq
m1: h=1 µm, 40 mΩ/sq
poly: h=0.5 µm, 5.71 Ω/sq
sub: h=0.3 µm, 50 Ω/sq

v23: h=0.9 µm, 3.6 Ω for 1 µm × 1 µm
v12: h=0.5 µm, 2 Ω for 1 µm × 1 µm
copoly: h=0.5 µm, 2 Ω for 1 µm × 1 µm
cosub: h=1.3 µm, 5.2 Ω for 1 µm × 1 µm

Figure 2: Cross section of the example process

Optionally, you can give a second number after the `tan delta` for the magnetic loss tangent:

```
layer   10   4.8   tan delta 0.002 0.004
```

The first layer given in the file is at the bottom of the layer stack, i.e., the stack is described in bottom-up order. The height of the last layer should be `infinity`.

You can also assign a name to a layer:

```
layer   0.3   7.2   name passivation
```

The name is only used for display by the `--print-process` option.

Each layer may contain conductors. The `conductor` keyword is followed by two numbers and a region expression. The region expression defines the areas where the conductor is present.

The units of quantities in the process file are usually specified implicitly through `assume` statements at the beginning of the file. In the example file, `assume microns` means that all the distances in the file are given in micrometers. If you want, you can specify the units for a particular quantity explicitly after the number. An example of that is at the end of the first `layer` line: the conductivity has been specified to be 5 Ωcm. Allowed units for lengths are `meters` (the default), `microns`, `angstroms`, and `kiloangstroms`. The conductivity of materials in layers, conductors, and vias may be specified in `Siemens/meter` (default) or `ohm-cm`. Conductor resistance can also be specified in `ohms/sq`. In IC processes, vias are often required to be a specific size, and the via resistance is given in Ω/via. EMX supports via conductivities given in `ohms/via` for rectangular vias. See the discussion of the `count` operator below for more information. Common abbreviations and alternate forms of units are also accepted, such as `um`, `kA`, and `S/m`.

Assume statements by default affect the units in all contexts, but you may also restrict the context. For example, suppose you have a process cross section where the units are in kiloAngstroms. Then it is natural to give most lengths in `kA`. However, within region expressions, which involve mask layout dimensions, the most convenient unit is usually micrometers. You could specify this as follows:

```
assume kA               # Affects all contexts
assume region microns   # Use microns within region expressions
```

The allowed contexts are `layer` (layer thicknesses and conductivities, `offset` and `position` statements), `conductor` (conductor thicknesses and conductivities), `via` (via conductivity), and `region` (lengths in region expressions). Some units, like `ohms/sq`, can apply only in one context. In those cases, an unqualified assumption affects only that context:

```
assume ohms/sq             # These two statements
assume conductor ohms/sq   # have the same effect
```

The vertical position of a conductor within a dielectric is determined from `offset` statements. Each `offset` gives a distance. To determine the position of the bottom surface of a metal, add up all the offsets between the `conductor` statement and the previous `layer` statement. This gives the distance from the bottom of the layer to the bottom of the metal. In figure 1, the metal `m2` is $0.6 + 1 + 1.5 = 3.1$ microns from the start of the 4.3 dielectric. *Note that the heights of the* `sub`*,* `poly`*, and* `m1` *metals do not affect this.* Note also that metals may extend through multiple dielectric layers. The top metal, `m3`, demonstrates this feature. You can also specify vertical positions with `position` statements. While `offset` specifies distance relative to the previous location, `position` is followed by a number giving the absolute distance from the start of layer in the last `layer` statement. For example, with the sequence `layer ...`, `offset 1.3um`, `position 1um`, the position is set to be 1 micron from the start of the layer.

Conductors can also have four optional attributes: `bias`, `color`, `name` and `internal bottom`. The attributes are specified after the `conductor` statement, like this:

```
conductor   1 um   3.7e-3 ohm/sq   metal2
  bias 0.03 um
  color purple
  internal bottom
```

The `bias` specifies an amount by which the outer edges of the conductor are "grown"; in this case, a 0.2 micron wire would become 0.26 microns wide in the final geometry. The `color` attribute is used by EMX only for displaying meshes (see the `--matlab-mesh` option in section 20). The `name` option is usually not required; by default, EMX uses the region expression (`metal2` in this example) as the name. If the region expression for the conductor is not just a single identifier, then you must give an explicit name in order to have a via contacting the conductor. An `internal bottom` declaration indicates that internal connections for this conductor should go on the bottom surface rather than the top (see the discussion under the `--bottom` option).

Vias between metals are specified with `via` statements. The statement includes two conductor names, a conductivity, and a layer expression. The via connects the named conductors in regions specified by the expression. One of the conductor names may be the special name `backside`, which refers to the ground plane at the bottom of the layer stack. In this case, the via acts as a physical ground connection for the other conductor. You can specify `color` and `name` attributes for vias just as you can for conductors.

For cases where you want to simulate a small piece of a larger layout, EMX supports an `emxmask` statement in the process file. The `emxmask` is followed by a region expression that bounds the part of the layout that is of interest. All of the other region expressions that define conductors, vias, capacitors, and resistors are automatically masked using the specified region. For example:

```
emxmask l30t0*l21t0
```

If the GDSII geometry is scaled during the fabrication, then you should specify the scaling factor with a `geometry scaling` statement like this:

```
geometry scaling 0.9
```

Scaling is applied when the GDSII file is read, and it only affects lateral dimensions, not vertical ones. For example, suppose the GDSII consists of wires with one micron width and separation, and the process file specifies that the wires are one micron thick. Then the statement above would give wires 0.9 microns wide and spaced by 0.9 microns, but still one micron thick. Operations in the process file (such as `grow`, `merge`, etc.) are done *after* the scaling is applied. Scaling was originally given by the deprecated `--scaling` command line option.

4.2 Geometric operations

Region expressions are built up from GDSII layers, like `L41T1`, and operators. The most common operators compute boolean combinations of regions (for example, intersections or unions). The recognized boolean operators are `+` (union), `*` (intersection), `^` (exclusive-or, or symmetric difference), `-` (difference), and `!` (complement). Examples of where such expressions are useful are given in the vias between `m1` and either `sub` or `poly`. In this case, there is one GDSII layer, `L48T1`, that codes both types of contacts. If the via is within `poly`, then it contacts `poly`; otherwise, it contacts `sub`.

Another useful operator is `grow`. The expression `grow(L49T1, 0.3um)` specifies whatever is in `L49T1` but grown outward by 0.3 microns. If the distance argument to `grow` is negative, it specifies a shrink. The use of `bias` for a conductor is equivalent to a `grow` by the bias amount. (The only difference is that EMX shows any nonzero biases whenever it prints a process cross section.)

The `merge` operator merges nearby regions: `merge(L20T0, 0.2um)` combines shapes on L20T0 that are within 0.2 microns of each other. This is the operation used internally within EMX to implement the `--via-separation` option discussed in section 20. However, if the standard via pitch is different for different conductors, then you may want to include the merging in the process file explicitly, like this:

```
    define via12 = merge(L15T0, 0.2um)
    define via56 = merge(L19T0, 0.5um)
```

Also, if a single GDSII layer encodes multiple types of vias, it is more efficient to do the merging initially. Consider these statements in the example process file:

```
    define cosub = L48T1-poly
    define copoly = L48T1*poly
```

If the standard via pitch on `L48T1` is 0.3 microns, then rather than specifying `--via-separation=0.3`, it is more efficient to change the process file as follows:

```
define contact = merge(L48T1, 0.3um)
define cosub   = contact-poly
define copoly  = contact*poly
```

There is also a three-argument version of `merge`. The third parameter is a distance that controls the "local reflex vertex" elimination step which merges vias that are diagonally offset. The default local reflex elimination distance is 2.5 times the merging distance. Setting the third parameter to zero will keep diagonal vias from merging. (Note that the third parameter is an absolute distance, not a scaling factor that is multiplied by the merge distance.) Also see the `--local-reflex-scaling` option.

The `count` operator is a special operator used when via resistance is specified in `ohms/via`. The via geometry is not affected by `count`, but it tells EMX that the layers mentioned in the argument represent vias that are to be counted when calculating resistance. In the above example, where `L48T1` may contact either poly or substrate, suppose that the vias resistance is 10 Ω/via for poly and 12 Ω/via for substrate. The `poly` in the region expression is used only as a mask, unlike `L48T1`, and is not required to consist of uniform rectangles. We indicate this by applying the `count` operator to `L48T1`:

```
define  contact = count(merge(L48T1, 0.3um))
define  cosub   = contact-poly
define  copoly  = contact*poly
via     sub  m1   12 ohms/via   cosub
via     poly m1   10 ohms/via   copoly
```

If via resistance is given in `ohms/via` and no `count` operator is used, the whole region expression is wrapped in an implicit `count`. This covers the most common case where no masking is required.

Some technologies support multiple via sizes, all with resistance given in `ohms/via`. These can be specified in EMX as follows:

```
assume microns
via  m1  m2  { 0.07 => 3.2 ohms/via,
               0.12 => 0.9 ohms/via,
               1.2e7 S/m }
     via12
```

This says that vias on `via12` of size 0.07 microns by 0.07 microns have resistance 3.2 Ω, while vias of size 0.12 microns contribute 0.9 Ω. And the technology also allows vias of general sizes (e.g., bar vias) whose resistance is calculated from the specified conductivity. (If the technology has rectangular vias, the number before the `=>` should be the square root of the product of length and width.)

By default, labels propagate through region expressions. So if you have a region expression involving `L48T0`, `L50T0`, and `L52T0`, then the resulting region will have a label wherever there is a label on any of the three input layers. You can suppress this automatic label propagation using the `nolabels` operator. For example, the expression `L48T0+nolabels(L50T0*L52T0)` has labels corresponding only to `L48T0`.

There are also two operators that are useful for advanced IC processes which have slotting rules for wide metals: `slotting` and `minslotting`. Both operators are designed to eliminate small holes that are electrically insignificant but that are required to satisfy design rules. To eliminate holes in `L50T1` that are less than 2 microns by 2 microns, use `slotting(L50T1, 2um)`. To eliminate holes that are no wider than 3 microns, but which may be longer, use `minslotting(L50T1, 3um)`.

For processes that require dummy metal fill to meet density requirements, you may want to run simulations with the metal fill eliminated. The `fill` operator removes small shapes that are less than a specified size. For example, if the metal fill consists of 3 micron by 3 micron squares, use `fill(L39T0, 3um)`.

The `arc` operation can be used to simplify circular arcs in the layout, approximating them with polygons. This can be helpful for reducing simulation time. The first argument to `arc` is a region expression, and the second is a number of degrees. A smaller number means a finer approximation (and less simplification). As an example, `arc(L10T0, 45)` will approximate arcs with segments spanning 45 degrees, e.g., circles will become octagons.

The `bbox` operator takes a region expression and returns a rectangle that bounds the corresponding region. If the argument represents an empty region, the bounding box is also empty. Similarly, if the argument represents the whole plane, then the bounding box is also the whole plane. The bounding box has no labels.

There are four selection operators: `interact`, `cut`, `inside`, and `outside`. Each takes a pair of region expressions, e.g.,

```
define m1in = inside(m1, m2)
define m1out = outside(m1, m2)
define m1rest = cut(m1, m2)
```

The selection operators take the region given by the first argument and decompose it into individual polygons. They then select those individual polygons that share area with the second region in different ways:

`interact` takes the polygons that have some area inside the second region;

`cut` takes the polygons that have some area inside and some area outside;

`inside` takes the polygons that have no area outside; and

`outside` takes the polygons that have no area inside.

The labels for a selection operation come only from the first region.

4.3 Definitions and arithmetic expressions

The example of figure 1 shows the use of `define` statements to give meaningful names to GDSII layers and to region expressions. You can also use `define` to give symbolic names to constants or to arithmetic expressions. You may use expressions and references anywhere EMX requires a numeric value. For example:

```
define metal1_bias = 0.1
assume microns
define metal1 = grow(L12T0, metal1_bias)
define metal2 = grow(L14T0, 1.1*metal1_bias)
```

The usual operators `+`, `-`, `*`, and `/` are available within arithmetic expressions. In this example, note that `metal1_bias` alone is dimensionless. Expressions only acquire a dimension when used in context. Here, within a `grow` statement, the expressions are interpreted as lengths.

Also available are the functions `exp` (exponential with base e), `log` (base e logarithm), `sqrt` (square root), and `pow` (x^y for two arguments):

```
define onehalf = sqrt(pow(0.5, 2))
define one = exp(log(1))
```

EMX also supports two types of conditional expressions. One is a `table` expression:

```
define m1bias = table (width)
   { 0.1 => 0, 0.2 => 0.05, 0.4 => 0.2 }
```

The argument in parentheses can be any numeric expression. EMX first evaluates the expression. If `width` is less than 0.1, the result is 0. If `width` is between 0.1 and 0.2, then EMX linearly interpolates between the 0 and 0.05 values. If `width` is between 0.2 and 0.4, then EMX linearly interpolates between 0.05 and 0.2. And if `width` is more than 0.4, the result is 0.2. EMX's `table` expressions are useful for transcribing tabulated process data, such as width- and spacing-dependent quantities. See section 14 for more information on such quantities.

The other type of conditional is an if-then-else:

```
define a = if(b < 5, b, 5)
```

The `if` operator evaluates the first argument. If it is non-zero, the result is equal to the value of the second argument; otherwise the result is the third argument. This particular example sets `a` to the minimum of `b` and 5. The arithmetic comparison operators `<`, `<=`, `>`, `>=`, `==`, and `!=` are useful for constructing conditions.

Definitions may be given in any order. Recursive definitions are not allowed, and identifiers may only be defined once.

Process files can also contain information about the statistical and temperature variations in the process. EMX uses this information for doing Monte Carlo and perturbation analyses. These features are described in sections 10, 11, 12, and 13.

```
⟨process file⟩→⟨declaration⟩∗
⟨declaration⟩→⟨layer⟩ ; ⟨position⟩ ; ⟨conductor⟩ ; ⟨via⟩ ; ⟨assume⟩ ;
    ⟨define⟩ ; ⟨capacitor⟩ ; ⟨resistor⟩ ; ⟨temperature⟩ ; ⟨scaling⟩ ;
    ⟨emxmask⟩ ; ⟨flipping⟩ ; unprintable
⟨layer⟩→layer ⟨length⟩ ⟨expr⟩ [⟨expr⟩ [⟨conductivity⟩]] ⟨layer opts⟩
⟨layer opts⟩→(name ⟨id⟩ ; conductivity ⟨conductivity⟩ ;
    tan delta ⟨expr⟩[⟨expr⟩])∗
⟨position⟩→(offset ; position) ⟨length⟩
⟨conductor⟩→conductor ⟨length⟩ ⟨conductivity⟩ ⟨expr⟩ ⟨cond opts⟩
⟨cond opts⟩→(bias ⟨length⟩ ; color ⟨color⟩ ; name ⟨id⟩ ;
    internal bottom)∗
⟨via⟩→via ⟨id⟩ ⟨id⟩ (⟨conductivity⟩ ; ⟨via conds⟩) ⟨expr⟩ ⟨col/name⟩
⟨via conds⟩→{ ⟨via conductivity⟩ (, ⟨via conductivity⟩)∗ }
⟨via conductivity⟩→⟨length⟩ => ⟨conductivity⟩ ; ⟨conductivity⟩
⟨col/name⟩→(color ⟨color⟩ ; name ⟨id⟩)∗
⟨assume⟩→assume [⟨context⟩] ⟨units⟩
⟨context⟩→layer ; conductor ; via ; region
⟨units⟩→⟨length units⟩ ; ⟨conductivity units⟩
⟨define⟩→define ⟨id⟩ = ⟨expr⟩
⟨capacitor⟩→capacitor ⟨id⟩ ⟨id⟩ ⟨expr⟩ ⟨col/name⟩
⟨resistor⟩→resistor ⟨id⟩ ⟨conductivity⟩ ⟨expr⟩ ⟨col/name⟩
⟨temperature⟩→temperature ⟨number⟩
⟨scaling⟩→geometry scaling ⟨number⟩
⟨emxmask⟩→emxmask ⟨expr⟩
⟨flipping⟩→begin flip ; end flip
⟨length⟩→⟨expr⟩ [⟨length units⟩]
⟨length units⟩→meters ; microns ; kiloangstroms ; angstroms
⟨conductivity⟩→⟨expr⟩ [⟨conductivity units⟩]
⟨conductivity units⟩→siemens/meter ; ohm-cm ; ohm/sq ; ohm/via
⟨expr⟩→⟨id⟩ ; ⟨number⟩ ; ( ⟨expr⟩ ) ; ⟨random⟩ ;
    ⟨unary op⟩ ⟨expr⟩ ; ⟨expr⟩⟨binary op⟩⟨expr⟩ ;
    ⟨function⟩ ( ⟨expr⟩ (, ⟨expr⟩)∗) ; ⟨table⟩
⟨unary op⟩→! ; -
⟨binary op⟩→+ ; - ; * ; / ; > ; >= ; < ; <= ; == ; !=
⟨function⟩→grow ; slotting ; minslotting ; merge ; nolabels ;
    count ; fill ; arc ; interact ; cut ; inside ; outside ; if ;
    log ; exp ; sqrt ; pow ; hide
⟨random⟩→random [⟨number⟩ *] sigma ⟨number⟩ [%]
    ⟨random opts⟩
⟨random opts⟩→(limit ⟨number⟩ ; scaling ⟨number⟩ ; ⟨corner⟩)∗
⟨corner⟩→corner ⟨id⟩ ⟨number⟩
⟨table⟩→table ( ⟨expr⟩ ) { ⟨table entry⟩ (, ⟨table entry⟩)* }
⟨table entry⟩→⟨expr⟩ => ⟨expr⟩
```

Figure 3: Process file syntax summary

4.4 Summary of process file syntax

Figure 3 is a summary of EMX's process file syntax. In this summary, syntax elements are enclosed in angle brackets $\langle \ldots \rangle$, keywords are shown like `this`, square brackets [...] denote an optional item, a semicolon ; separates alternatives, and a star $*$ denotes zero or more repetitions of an item. For example, the first few lines in the figure have the following meaning:

1. the process file consists of zero or more declarations;
2. a declaration can be either a layer, a position, a conductor, etc;
3. a layer consists of the keyword `layer` followed by a length, an expression, an optional expression and conductivity, and some layer options.

The process file is usually interpreted in a case-insensitive manner; for example, identifiers `VIA23` and `via23` are considered the same. You can give the option `--case-sensitive` to preserve the case of identifiers, though keywords such as `layer` or `conductor` are always recognized regardless of case. An identifier $\langle$id$\rangle$ must start with a letter, and may contain letters, digits, and underscores. Identifiers must be distinct from keywords. In addition, the following identifiers have predefined special meanings in EMX: `deltat`, `dtemp`, `backside`, `frequency`, `width`, `spacing`, `infinity`, and identifiers like `L13T0` that have the form of GDSII layers.

Numbers begin with either a digit or a decimal point. You can also use scientific notation: `1.23e-7` means 1.23×10^{-7}. Recognized color names can be found using the `--list-colors` option.

5 2D and 3D conductors

EMX normally makes a 2D approximation to the conductors. In this approximation, the current flow in the metal is distributed throughout the volume, but charge can only accumulate on the top and bottom surfaces of the metal. Further, the charge on the two surfaces must be identical. Any capacitive effects due to the conductor sidewalls are ignored.

You may use the `--3d` to request full 3D models of the conductors. In the 3D model, the current flow is still distributed throughout the volume, but charge may accumulate on any surface, and all the charges are independent. The current flow may also be split vertically into multiple layers using the `--thickness` option. Both `--3d` and `--thickness` are discussed in section 20.

6 Ports

Labels in the GDSII layout specify locations for ports. Most commonly, the label for a metal will be on the same GDSII layer as the metal. However, if the labels are on a different layer, you can use a region expression to combine the two. For example, if `L30T0` represents metal 1, but the labels for metal 1 are on `L30T10`, then you can use `L30T0+L30T10` as the region expression for metal 1. This will pick up both the geometry and the labels.

The default type of port in EMX is a voltage source attached to an edge in the layout. The source supplies current that flows across the edge. Note that if the edge is very long, current will be injected over a very large region; this is often not what you want. For example, suppose the layout is a square parallel-plate capacitor. If you put a label at some point on the edge of the top plate, then current will flow into the capacitor across that entire edge, not just from the area near the label. Instead, attach a small stub to the plate and put the port label on the stub. Then the current will flow through the stub and spread out from the area of the label to the plate.

The default source formulation allows current to flow from one part of a port connection to another via a zero-impedance return path. If there are multiple mesh elements within a connection region, then the intra-source loops that are formed can act as small parasitic inductances. The effect of these parasitic loops may be noticeable in some cases, e.g., in wide high-Q inductors with low inductance. The option `--uniform-sources` eliminates these parasitic loops; current is then forced to flow uniformly into (or out of) each connection region.

EMX also supports ports that are placed in the interior of a conductor. In this case, the voltage source supplies current that flows into a small region around the port location. You can declare the labels for interior ports with the `--internal` option, described in section 20. Or, if you are using EMX with the Cadence layout tools, you can use the `--cadence-pins` option to get the internal ports from information embedded in the GDSII. Internal ports connect at the top of the conductor unless overridden by a `--bottom` option or by an `internal bottom` declaration in the process file.

By default, the other end of a port's voltage source is grounded. Ground represents the infinite ground plane at the bottom of the layer stack. However, sometimes you may wish to consider ground to be some other piece of metalization. For example, if you are simulating a test structure for wafer probing, the test structure will have ground and signal pads, and you may want to consider the ground for each signal pad to be the associated ground pad. You can specify this in EMX using the `--port` option. Different ports can have different associated grounds.

EMX sorts the port names lexicographically to compute the port ordering for all output files. For example, if the port names are `p1`, `p2`, ..., `p10`, the output order will be `p1`, `p10`, `p2`, ..., `p9`. So if you have more than 9 (or 99, or 999) ports and want the sorting to be numeric, be sure to include appropriate padding: `p01`, `p02`, ...

Sometimes you can get better accuracy by specifying the desired driving modes for the ports. As an example, consider a symmetric center-tapped inductor. In normal operation, the center tap is at AC ground, and the two input ports are driven differentially. The normal differential operation is obtained as the superposition of two modes: one with the first input port driven and the center tap and second input grounded, and one with the second input port

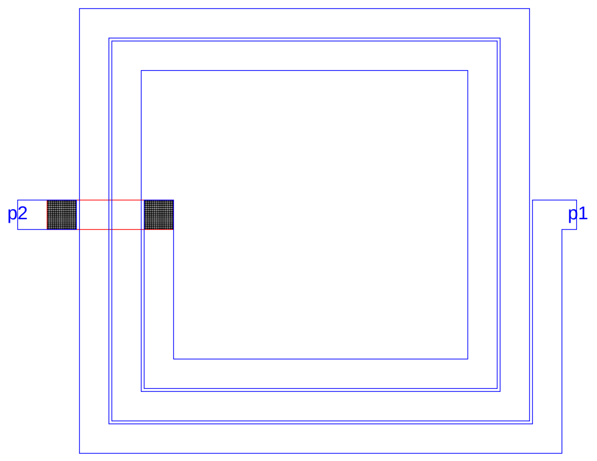


Figure 4: Example inductor layout

driven and the tap and first input grounded. These two modes usually have a significant current flowing out of the center tap, but this current largely cancels out in the differential-mode superposition. As a result the remaining differential current is not determined as accurately as it would be from a single simulation with the ports driven differentially. You can tell EMX to do a differential-mode simulation using the `--mode` option, like this:

```
emx ... --mode=+p1-p2 ...
```

EMX automatically adds appropriate orthogonal modes to whatever modes you specify. In the example above, the common mode with `p1` and `p2` driven in-phase would also be simulated. This option does not change the form of the final output, i.e., the Y- or S-parameters will be approximately the same as without `--mode`. However, they are derived internally from differential-mode and common-mode simulations, and hence are more accurate when played back under differential stimulation.

7 An example

An example inductor layout is shown in figure 4. This layout is available in the file `exind.gds`. The inductor uses the process file shown in figure 1, which is available as file `exproc.proc`. The blue layer is the top metal, `m3`, the red is `m2`, and the small black squares are vias on `L52T1`. Note the labels on `m3` defining ports `p1` and `p2`.

To begin, run EMX with a low accuracy setting:

```
./emx --edge-width=2 exind.gds ind exproc.proc 5e9
```

There's an option here, `--edge-width=2`, that controls EMX's discretization of the structure. It tells EMX to use elements at the metal edges that are about

2 microns wide. If you enter this command, EMX should run for a few seconds and then print this set of Y-parameters:

```
Frequency 5.000000e+09:
            p1                   p2
 p1  2.78e-03-1.99e-02j -2.02e-03+2.15e-02j
 p2 -2.02e-03+2.15e-02j  2.83e-03-1.99e-02j
```

Now run EMX with a higher accuracy setting to check the result:

```
./emx --edge-width=0.75 --3d=* exind.gds ind exproc.proc 5e9
```

The `--3d=*` option tells EMX to treat all metals as three-dimensional. The output in this case is the following set of Y-parameters:

```
Frequency 5.000000e+09:
            p1                   p2
 p1  2.95e-03-1.97e-02j -2.11e-03+2.14e-02j
 p2 -2.11e-03+2.14e-02j  3.01e-03-1.97e-02j
```

Compared to the low accuracy simulation, the effective inductance (the imaginary part of $1/Y_{11}$, divided by 2π times the frequency) has changed by 1%. The resistance (the real part of $1/Y_{11}$) has changed by 7%, and the Q at this frequency has changed by 7%. Further refinement does not change the answer much for this example.

8 Capacitors

Thin-film parallel-plate capacitors require special handling in EMX. In such capacitors, the plate separation is very small compared to the size of the plates. The significant charge distribution is mostly between the plates, and inaccurate answers will result unless the mesh elements align with the plate edges. To simulate structures containing thin-film capacitors accurately, you must tell EMX where the capacitors are. To do this, add a line to the process file as shown in figure 5. The `capacitor` statement specifies two conductor names and a region expression. The area defined by the region expression will be treated specially by EMX when creating the mesh.

9 Resistors

EMX supports conductors with different conductivities in different areas, e.g., a polysilicon layer that can be doped in certain areas to alter the conductivity and form resistors. To specify such resistors, you use a `resistor` statement in the process file (figure 6). The statement gives the conductor, an expression for the conductivity, and an expression defining the region of the resistor. *Note that the conductor area is not affected by the resistor region (i.e., the region is masked by*

```
assume microns
assume ohm/sq

define m1  = L1T1
define cap = L2T1  # Capacitor top plate
define m2  = L3T1

layer           300.00     11.9   1   10 S/m
layer             5.00      3.9   1    0
        offset    1.30
        conductor 1.00  30e-3   m1
        offset    1.00  # Now at the top of m1
        offset    0.03
        # The gap between m1 and cap is 0.03um
        conductor 0.20 150e-3   cap
        offset    0.70
        conductor 1.00  30e-3   m2
layer               1      7.0   1        0
layer        infinity      1.0   1        0

# There is a thin-film capacitor between
# m1 and cap in the region where they
# overlap (m1*cap)

capacitor  m1  cap  m1*cap
```

Figure 5: Process file with thin-film capacitor

```
assume microns
assume ohm/sq
assume ohms/via

define cont = L5T0
define poly = L8T0
define mt1 = L10T0
define rpo = L13T0  # resistor doping region

layer   150.0  11.9  1  10 ohm-cm  name sub
layer     0.4   3.9                name fox
layer     0.8   4.2                name ild
conductor 0.2  10.0  poly  # normal 10 ohm/sq
layer     0.5   3.9                name imd
conductor 0.4   0.1  mt1
layer     0.5   2.0                name pass1
layer infinity  1.0                name air

via  poly  mt1  8.2  cont

# poly is doped differently within rpo
resistor poly 230.0  rpo   # 230 ohm/sq res
```

Figure 6: Process file with doped resistor

the conductor). If the conductivity expression for the resistor involves geometry-dependent values (see section 14), then those values refer to the geometry of the conductor, not the resistor region.

10 Monte Carlo analysis

EMX can simulate the effect of process variation on a structure using Monte Carlo analysis. For a Monte Carlo analysis, EMX runs multiple simulations with different random perturbations of the process parameters and produces separate output files for each simulation.

Before running a Monte Carlo analysis, you need to augment the process file with information about the allowed statistical variations. An example process file with statistical variation is shown in figure 7. A random variable is declared with the `random` keyword. All random variables are assumed to have a Gaussian, or normal, distribution with average value zero. The expression following `random` defines the standard deviation.

In the example, there are three random variables. For the first variable, a 10% variation represents three standard deviations (`3*sigma`). Equivalently, you could specify one standard deviation using `sigma 3.33%`. The second variable

```
assume microns
assume layer ohm-cm
assume conductor ohms/sq

# One standard deviation is 3.33%
# Default limit of 3 standard deviations
define dh = random 3*sigma 10%

# m1 bias is restricted to 0.1 +/- 2*0.15 [-0.2, 0.4]
define m1 = grow(L1T0, 0.1+(random sigma 0.15 limit 2))

layer  200          11.9   1   30+(random sigma 5)
layer    5+dh       4.2
  offset 2+dh
  # Note that sheet resistance uses - instead of +
  # When dh is positive, the conductor is thicker, and the
  # sheet resistance is lower
  conductor   1.5+dh   40e-3-dh   m1
layer    1+2*dh    7.1
layer  infinity    1.0
```

Figure 7: Process file showing Monte Carlo features

has a standard deviation of 0.15, and the last a standard deviation of 5. When EMX runs a Monte Carlo analysis, each variable will be assigned a new random value at the start of each individual simulation. Note that quantities which use the same named random variable, like `dh`, will be correlated. In the example, if `dh` gets the value 2.3%, then all the layers will be stretched by 2.3% of their nominal height, except for the top layer, which will increase by 4.6%. The other (anonymous) random variables, which specify the metal bias and substrate resistivity, are completely uncorrelated with `dh` and with each other. Variations may be either relative (specified as a percentage) or absolute numbers.

Perturbations may be applied to any of the numbers in the process file. Perturbations themselves are dimensionless; how they are interpreted depends on the context in which they are used. In the example, `dh` is used both as a length and as a sheet resistance. If you want to specify explicit units for a quantity, the units should be placed after the perturbations. Numbers may have more than one perturbation, e.g., `3+2*dh-dt microns`.

Random variables may also have an optional `limit`. The limit is in numbers of standard deviations. EMX will restrict the random choice for each variable to lie within the specified number of standard deviations of zero. The reason for allowing a limit is that in the manufacturing process, devices where the parameters fall far outside of the nominal values will be rejected, so the behavior of those devices should be ignored. Also, limiting the random variations may be used to ensure that the simulated geometry is always valid. In the example, the conductor bias is restricted to ensure that adjacent conductors at minimum spacing cannot merge together. The default limit is three standard deviations.

Normally EMX sets all random variables to zero when running a simulation. There are three ways to get EMX to choose non-zero values. The first is to specify the `--monte-carlo=`*number* option. This tells EMX to run the specified number of simulations, typically producing a series of output files. The second is to specify `--random-seed=`*number*. The choice of random values with `--random-seed=`*n* is the same as in the *n*th simulation done with `--monte-carlo`. This allows you to replay an individual simulation from a Monte Carlo analysis, or to see the process file (using `--print-process`). The final way is to do a perturbation analysis, discussed in the next section.

EMX uses the same meshing parameters for all the simulations in a Monte Carlo analysis. If you use a random conductor bias, then the mesh, and possibly the answer, can be significantly affected by tiny changes in the bias. You should ensure that the `--edge-width` is set low enough that the narrowest conductors are adequately meshed.

11 Perturbation analysis

Often the effects of process variation are small enough that they may be approximated by a linear model. Suppose for example, that we are concerned with the statistical variation in the Q of an inductor. Q is given by $-\operatorname{Im}(Y_{11})/\operatorname{Re}(Y_{11})$. Assume that there are two independent random perturbations, δ_1 and δ_2, in the process file. Then if δ_1 and δ_2 are small enough, we may expand Y_{11} in a Taylor series:

$$Y_{11}(\delta_1, \delta_2) \approx Y_{11}(0,0) + D_1(Y_{11})(0,0)\delta_1 + D_2(Y_{11})(0,0)\delta_2$$

For conciseness, define $d_i = D_i(Y_{11})$, and write Y_{11} for $Y_{11}(0,0)$, d_1 for $d_1(0,0)$, etc. Pushing derivatives through the definition of Q gives:

$$Q(\delta_1, \delta_2) \approx Q - \frac{\operatorname{Im}(d_1\delta_1 + d_2\delta_2)}{\operatorname{Re}(Y_{11})} + \frac{\operatorname{Re}(d_1\delta_1 + d_2\delta_2)\operatorname{Im}(Y_{11})}{\operatorname{Re}(Y_{11})^2}$$

We may write this as:

$$Q(\delta_1, \delta_2) \approx Q + D_1(Q)\delta_1 + D_2(Q)\delta_2$$

If δ_1 and δ_2 are independent and have Gaussian distribution with standard deviations σ_1 and σ_2, then $Q(\delta_1, \delta_2)$ is Gaussian with standard deviation

$$\sqrt{(D_1(Q)\sigma_1)^2 + (D_2(Q)\sigma_2)^2}.$$

In a perturbation analysis, EMX computes the derivatives of each output file with respect to each independent variable in the process file. These derivatives may be used to compute the statistical properties of performance functions like Q as illustrated above. The derivatives are stored in auxiliary output files. Each auxiliary file has the same format as the corresponding output file. *The* *derivatives for a variable are scaled by the standard deviation of that variable.* In example above, there would be two auxiliary files, one containing $D_1(Y_{11})\sigma_1$, $D_1(Y_{21})\sigma_1$, $D_1(Y_{12})\sigma_1$, etc., and the other containing $D_2(\ldots)\sigma_2$. You instruct EMX to run a perturbation analysis using `--perturbation`.

By default, EMX uses simulations where the process parameters vary by one standard deviation for the perturbation analysis. You can change the variation for an individual variable using an optional `scaling`. For example,

```
define dh = random sigma 10% limit 2 scaling 0.5
```

will make EMX vary `dh` by half a standard deviation during perturbation analysis. As with Monte Carlo analysis, make sure that the `--edge-width` is set low enough that the narrowest metals are always meshed sufficiently.

When choosing between Monte Carlo analysis and perturbation analysis, consider these tradeoffs:

- Monte Carlo analysis requires a large number of simulations (typically hundreds) to build up an accurate knowledge of the statistical distribution. Perturbation analysis requires work proportional to the number of independent random variables in the process file. It is much faster when the number of independent variables is small.
- If small variations in the process parameters change the device behavior in a strongly nonlinear manner, then perturbation analysis is inaccurate.

Note that you can also generate pseudo-Monte Carlo results from the perturbation analysis by generating random deviates, scaling the derivative files, and adding the results to the nominal simulation. This may be the easiest and fastest way to visualize the possible variations if the performance functions you are interested in are not easily differentiated.

12 Corner cases

EMX also lets you specify multiple corner cases within the process file. Any random variable may be given a set of possible fixed values using `corner` declarations. For example, consider this fragment:

```
define dr = random 3*sigma 10%
            corner lowres -10   # -10 percent
            corner highres +10  # +10 percent
...
conductor   1.2um   55e-3+dr ohms/sq   me7
```

This defines a varying sheet resistance for `me7`. You can fix the resistance to the low 3σ limit on the command line:

```
emx ... --corner=lowres
```

Similarly, specifying `highres` for the corner will set the sheet resistance to the upper limit.

Multiple `--corner` options may be specified on the command line, but EMX will print an error message if any individual random variable has more than one applicable `corner`.

13 Temperature

EMX process files can contain information about temperature dependencies. The numbers given in the process file represent values at some nominal temperature. The nominal temperature is specified by a `temperature` line in the process file:

```
temperature 35
```

and you specify the simulation temperature with the `--temperature` option:

```
emx ... --temperature=100
```

By default, EMX assumes that you are using degrees Celsius for the temperature, and it uses a nominal temperature of 25 C.

You can use the two pre-defined variables `deltat` and `dtemp` to model temperature dependencies. Both variables are computed by taking the difference between the value of the `--temperature` option and the process file temperature. The distinction between the variables is that `deltat` represents a percentage, while `dtemp` is just a number. For example:

```
conductor   3um   20e-3+0.4*deltat ohms/sq   metal4
```

This says that `metal4` has a temperature coefficient of 0.4% per degree (note the percent sign). This is equivalent to:

```
conductor   3um   20e-3*(1+0.004*dtemp) ohms/sq   metal4
```

For a simple linear temperature coefficient you can use `deltat`, but for second-order coefficients or other more complicated temperature dependencies, you have to use `dtemp`.

14 Geometry-dependent values

EMX allows geometry-dependent quantities for the conductor sheet resistance (conductivity), the conductor `bias` attribute, and for the second argument of the `grow` operator. Within such quantities, the special variables `width` and `spacing` refer to the geometry's local width and spacing *in microns*. Be especially careful with units when using `width` and `spacing`. In particular, this:

```
   assume meters
   define m1=grow(l30t0, 0.1*width + 0.1e-6)
```

almost certainly does *not* do what you want. Since `width` is always in microns, a one micron wide wire would be grown by $0.1 \times 1 + 10^{-7} = 0.1000001$ *meters*. One correct way to grow a wire by 10% plus an additional 0.1 micron would be:

```
assume meters
define m1=grow(l30t0, 0.1*width+0.1 um)
```

EMX will not warn you about confusion of units.

The local width and spacing are not easy to define precisely, but roughly are given by the sizes of the biggest circles that fit entirely inside and outside of the geometry, respectively. For a uniform array of wires, these quantities have the intuitive meanings.

As an example, suppose that the bias of a wire is 10% of the width of the wire, up to a maximum of 0.15 microns. You can define this by:

```
assume microns
define m5bias = if(0.1*width > 0.15, 0.15, 0.1*width)
conductor   0.5   10e-3 ohm/sq   m5
  bias m5bias
```

Note that variable `bias` values (or `grow` amounts) can result in simple rectangles turning into odd-looking shapes near bends, corners of other shapes, etc. In order to keep the mesh complexity under control, EMX does not allow the bias amount to change too rapidly. Instead, EMX computes a single average bias for line segments up to a few microns in length and breaks long segments up into shorter segments of such size.

Computing geometry-dependent quantities during mesh generation is time-consuming. For simple structures such as interdigitated capacitors with uniform widths and spacings, you can specify the width and spacing directly from the command line with the `--width` and `--spacing` options. This has the effect of fixing the width and spacing computations to always return the specified values. These options are also useful with the `--print-process` option if you want to check the process file for various width and spacing combinations. If you do not specify these options, then `--print-process` uses the first case in `table` and `if` expressions.

15 Frequency-dependent values

Layer properties (permittivity and permeability, loss tangent, and conductivity) can be frequency-dependent in EMX. You specify the dependence by giving an expression that involves the special variable `frequency`, which is in Hz. For example:

```
layer   10um   5.2-0.2*(frequency/1e9-5)   tan delta 0.013
```

This gives a relative dielectric constant of 5.2 at a center frequency of 5 GHz, decreasing by 0.2 per GHz as frequency increases. Conductor and via resistance is also allowed to be frequency-dependent.

When doing a sweep, EMX needs to be able to approximate the Green's function by a polynomial over the range of sweep frequencies. Discontinuities or rapid changes in the layer properties may cause EMX to print a message about the Green's function interpolation requiring too many frequencies. That message indicates that you should check the frequency-dependent expressions to ensure that they specify physically appropriate values over the required frequency range.

If you want EMX to print the process at a particular frequency, specify a GDSII file, a cell name, and the frequency along with the process file and the `--print-process` option. The GDSII file and cell name are not used in this case, so it's OK if the GDSII file or the cell don't exist. By default `--print-process` uses the maximum frequency specified on the command line, or 1 Hz if no frequency is given.

16 Encryption

Some foundries that support EMX prefer not to distribute plain-text process files. Instead they provide a fully or partially encrypted process, along with a key. Using such a process file with EMX requires you to specify the key using the option `--key`.

If you wish to create your own encrypted process files, you'll need to use one of the options `--encrypt-process` or `--partial-encrypt-process`. The first option creates a fully encrypted file and is the simplest. When you run:

```
emx --encrypt-process vlsi.proc --key=secretagent
```

then EMX will create a binary file called `encrypted.proc` which contains the information from `vlsi.proc`. The end user will need to supply the option `--key=secretagent` when running EMX with the encrypted file, but EMX otherwise acts just as if `vlsi.proc` had been used.

If multiple process files are required (supplied via the `--definitions-file` option), then all the files must be specified at encryption time, and all the information is merged together into `encrypted.proc`. The end user will need only the one encrypted file.

Note that EMX's `--print-process` option can still be used to display process information. If you want to disable printing of information like dielectric constants, thicknesses, etc., you can add a single line containing `unprintable` at the end of the process file before encrypting. The process display will then show a picture of the process cross section, but no detailed physical information will be included.

Using partial encryption is more flexible, in that you can encrypt only desired parts of the process file while other parts remain visible. This allows the end user some flexibility in modifying the file; for example, they might be able to add extra layers and conductors to a VLSI stack in order to model additional flip-chip processing (see also section 17). Here's an example a section from a process file that's intended for a partial encryption:

```
...
layer <<<hide(0.05)>>> <<<hide(4.7)>>> name ILD3a
layer <<<hide(0.12)>>> <<<hide(3.5)>>> name ILD3b
conductor <<<hide(0.12)>>>  0.085 ohms/sq  metal3
...
```

There are two important things to note in this example. The first is the use of the `<<<` and `>>>` brackets. When EMX encrypts a process file with `--partial-encrypt-process`, anything between such brackets gets turned into a binary blob that is represented in the encrypted file as a string of hexadecimal digits. For this example, `encrypted.proc` will contain lines like this:

```
...
layer encrypted f3...7axx encrypted b2...cfxx name ILD3a
layer encrypted 21...69xx encrypted 75...a8xx name ILD3b
conductor encrypted 21...69xx  0.085 ohms/sq  metal3
...
```

The second point to note is the use of the `hide` function in expressions. This doesn't change the value of the expression, but it does tell EMX that the expression should not be displayed in the process cross section. Any information that depends indirectly on a hidden value is also suppressed. For example, EMX normally displays the conductor sheet resistance in Ω/sq in the cross section. If you changed the `0.085 ohms/sq` to `1e7 S/m`, then EMX would internally calculate the sheet resistance from the conductivity and the conductor thickness. However, this sheet resistance would not be displayed in the cross section since it depends on the hidden thickness value.

When you are developing a process file that is intended to be distributed after partial encryption, you can still run EMX on the unencrypted file. Occurrences of `hide` and `<<<` and `>>>` are effectively ignored.

17 Flipping layers

In order to make it easier modify process files for use in situations like flip-chip arrangements, EMX offers a way to invert a group of layers and conductors. Here is an example:

```
...
begin flip
layer  2  3.9  name IMD1
conductor  1.5  3.3e7 S/m  M1
layer  0.2  7  name IMD1a
layer  3  3.9  name IMD2
conductor  2.5  3.3e7 S/m  M2
end flip
...
```

The three layers `IMD1`, `IMD1a`, and `IMD2` will be inverted in the process, and `M2` will appear below `M1`. Flipped regions can be nested.

18 Creating models

EMX includes an experimental facility for producing pole-zero (state-space) models. These are useful when you want to run transient analysis in a circuit simulator, since simulators often have problems when running transient simulations involving S-parameter devices. The accuracy of the model is not guaranteed, but the response is usually fairly close to the frequency-domain simulation data. While the model includes controlled current and voltage sources, the format of the model guarantees that it is stable and passive, and that the noise behavior is correct. Model creation requires that you be running a frequency sweep. Typically you would start the sweep at DC (frequency zero), though this is not required; if you do not start at DC, then the model will be an open circuit there.

Internally, EMX uses two model creation algorithms. The first is a general method whose time complexity grows rapidly with the number of ports. It is reasonably quick for most devices with a few ports, but around ten ports is probably a practical limit.

The second method is suitable only with processes where the losses come solely from the conductors. There should be no dielectric conductivities, no (or only very small) loss tangents, and no frequency-dependent layer properties. This method must be explicitly requested using the `--model-reduce-only` option. It is significantly faster and can handle much larger numbers of ports, but it usually produces models that are slightly larger than the first (general) algorithm.

In both cases, the size of the final model grows quadratically with the number of ports and with the number of poles needed to capture the response. A model with tens of ports that needs two or three hundred poles can easily have tens of thousands of elements in the generated netlist, so use with care.

The output file for the model is given by the `--model-file` option. You can save models in various formats; see subsection 20.9 for details.

The objective function for model creation is to try to match S-parameters. If the metrics you are interested in are sensitive to slight changes in the S-parameters, you may need to tighten the default modeling tolerance in order to get an accurate fit. Typical examples include things like the high Q of an inductor or a weak coupling between separated structures. See subsection 20.9 for options to control the model accuracy. You can also influence the objective function via `--mode` options. The standard S-parameters are converted to mixed-mode S-parameters for fitting when you specify the port driving modes. This may help the accuracy slightly when you're interested in weak couplings.

19 Black boxing devices

If your layout includes subcells that you would like to exclude from the simulation, EMX can cut out those cells and add ports at their terminals. The top-levels labels (or Cadence pins, if you're using `--cadence-pins`) in the excluded cells will be renamed to labels of the form `~x0000`, `~x0001`, etc. These labels or pins then attach to the remaining layout in the usual way (e.g., labels snap to nearby edges) and become extra ports. These extra ports are automatically included in the simulation (even if you don't list them using `--include`). The simulated electrical parameters can then be stitched together with models of the excluded devices to form a schematic for the whole layout.

You specify the devices to exclude using the `--device-cells-file` option. The file should contain a list of regular expressions, one per line. Any cell whose name matches one of the regular expressions will be black-boxed. When EMX finishes the simulation, it will write an output file called `emxdevices.txt` that contains the information on the excluded cells. An example output file is shown in figure 8. The first number is the number of device instances (three in the example). Each instance has the name of an excluded cell, its coordinates in the layout, the number of ports for the device, and how the ports were connected. In the example, the first record says that a cell `nmos1` with three connections was excluded. The first connection was called `D` in the cell's top level, and it corresponds to port `~x0002`. The second connection was `G`, and it corresponds to port `~x0000`. Note that when the same black-box cell is used multiple times, then there is a separate record in the `emxdevices.txt` file for each such instance. Following the devices instances is another number (12 in this case) representing the number of ports (both automatically generated and external). The last set of lines is the list of ports. Each port is followed by a number (0, 1, or 2). Regular ports that were solved for have the number 2; the number of ports in the S-parameter or Y-parameter output is equal to the number of regular ports. Ports with the number 0 (`VSS` in this example) were grounded during the solve. Ports labeled with 1 (`~x0001` and `~x0004`) were isolated device terminals. These were also not solved for, but their connectivity is still needed to generate a final schematic. In this case we see that the `S` terminal on the first `nmos1` device was connected directly to the `D` terminal of the second `nmos1` device, and that the `S` terminal of the second `nmos1` connected directly to `P1` of `cap350f`.

When using the black box functionality, there are two important caveats for your layout style. First, you must ensure that you route to places on the excluded devices so that the cell's labels and pins connect correctly. Second, be careful that you are not relying on any of the internal structure of the excluded cells for connectivity. As a simple example of the possible danger, suppose you have an excluded device that has a label on `metal3` representing a terminal, and there are two (non-device) `metal3` paths that route to that terminal. If the two paths are disjoint, then the automatically-generated label for the terminal will snap to one of them, and the other path will be left floating. The correct set up would be to route the two paths together, and then have a single connection to the device.

```
3
nmos1 5.79 -167.515 3
D ~x0002
G ~x0000
S ~x0001
nmos1 5.79 -163.035 3
D ~x0001
G ~x0003
S ~x0004
cap350f 40.75 -163.825 3
GND ~x0006
P1 ~x0004
P2 ~x0007
12
RFIN 2
RFOUT 2
VDD 2
VSS 0
~x0000 2
~x0001 1
~x0002 2
~x0003 2
~x0004 1
~x0005 2
~x0006 2
~x0007 2
```

Figure 8: Example emxdevices.txt output file

20 Controlling EMX

One of the most common things you'll want to do is to simulate a structure at multiple frequencies. You can do this with EMX by just giving the list of desired frequencies on the command line. This is essentially the same as invoking EMX multiple times, but it's much faster since some parts of the computation are frequency-independent and can be shared. As an example, if you want to simulate your layout at 1, 2, 3, and 4 GHz, use this command:

```
emx layout.gds mystructure foundry.proc 1e9 2e9 3e9 4e9
```

(If you just want the response within a frequency range, you can use the `--sweep` option, discussed below, to get an automatic frequency sweep.)

You'll probably want to specify additional options to control EMX. Options begin with two dashes (`--`). All options may be abbreviated as long as the result is not ambiguous (e.g., `--via-sidewalls` and `--via-side` are considered to be the same). Some options require an argument, such as a number or name. The argument follows `=`, e.g., `--edge-width=2`. Some options have single-letter forms which are preceded by one dash (e.g., `-e 2` instead of `--edge-width=2`). For the single-letter forms, the option's argument is separated from the option with a space. The options are classified here according to their primary function.

20.1 Meshing

`--edge-width=`*number* EMX meshes the structure so that the elements near the edge have a width given by this value. Generally you should set this to be on the order of the skin depth at the highest frequency of interest. EMX also constrains the maximum size of a shape in any dimension to be about ten times the edge width. You may abbreviate this option as `-e`. The number is in microns, and the default edge width is 1 micron.

You can override the edge width for an individual layer (either a conductor or via) using the form `--edge-width=`*layer,number*. The default edge mesh for a via is the geometric mean of the edge meshes for the two layers connected by the via.

`--3d=`*layer list* By default, EMX uses 2D approximations for all conductors. Specifying `--3d` tells EMX to instead use 3D models for any listed conductor. The layer list has one of the following forms:

`*` All metals are 3D.

`me1,me5,me6` Metals `me1`, `me5`, and `me6` are 3D.

`-*` No metals are 3D.

`-me1,me2` All metals except `me1` and `me2` are 3D.

The abbreviation for `--3d` is `-3`.

`--sidewalls=`*layer list* For wide 3D conductors, the sidewall capacitance sometimes has little effect. You can tell EMX to omit the sidewalls in order to save some time and memory using the `--sidewalls` option. By default, all 3D conductors have sidewalls, so the only useful form of this option is to give a list of conductors whose sidewalls should be omitted. For example `--3d=me5,me6 --sidewalls=-me5` requests two 3D conductors, but only `me6` has a sidewall.

`--via-sidewalls=`*layer list* Normally, EMX accounts only for via resistance. This option tells EMX to also model via capacitance. The list of vias has the same form as the list of metals for the `--3d` option.

`--via-inductance=`*layer list* This option makes EMX include the effects of via inductance for the specified layers. The list of vias has the same form as the list of metals for the `--3d` option.

`--via-edge-factor=`*number* EMX meshes vias so that elements near the edge of a via have a width given by this number times the edge width for conductors. Most early versions of EMX had this set at 4, but the default is now 1.

`--max-edge-factor=`*number* EMX normally uses bigger mesh elements on the inside of conductors than on the edges, and on the edges, it makes the elements longer in the direction parallel to the edge. The maximum length for elements is given by the `--max-edge-factor` value times the `--edge-width` value. The default is 8, but in some cases a smaller value is warranted. The most common case is with interdigitated capacitors with very thin fingers, where to otherwise limit the length of the mesh elements in the fingers would require an unreasonably small edge width.

`--thickness=`*number* This option provides a way to split thick metals vertically. In other words, it operates something like `--edge-width`, but in the vertical direction. Only 3D metals (and vias with sidewalls) can be split. However, for compatibility with earlier versions of EMX, if `--thickness` is specified but not `--3d`, then any conductors of more than the given thickness are treated as 3D, but without sidewalls. You can abbreviate `--thickness` with `-t`.

The thickness for an individual metal may be overridden using the form `--thickness=`*layer,number*.

`--max-splits=`*n* This limits the number of vertical splits for any metal. It is useful when there are both thick and thin metals, and you want to split the thin metals without splitting the thick metals too much.

`--via-separation=`*number* IC design rules often require the use of arrays of small vias instead of single large vias for connecting metals. This leads to a lot of wasted mesh elements. To avoid this waste, EMX merges vias that are “close,” forming a single large via. Two vias are considered close if they are within the distance specified by this option. The default is 0.5 (microns). If you don't want to merge vias, specify 0. This option may be abbreviated by `-v`. Via merging can be controlled on a layer-by-layer basis in the process file.

`--local-reflex-scaling=`*number* Merging diagonal lines of vias requires a "local reflex vertex" elimination step. The two-argument `merge` operation uses a local reflex distance equal to a certain scaling times the merge distance. This option controls the scaling; the default is 2.5. You can control the local reflex distance explicitly for an individual layer by using the three-argument form of the `merge` operation.

`--scaling=`*number* This option is equivalent to specifying a `geometry scaling` in the process file. Use of this option is deprecated.

`--excluded-cells-file=`*name* Omit any cells in the GDSII file whose name matches one of the regular expressions in the specified file. The file should have one regular expression per line. The allowed regular expression syntax follows that of the Perl programming language. Example regular expressions:

`npn4` A cell named `npn4`

`npn.*` Any cell whose name begins with `npn`

`mosfet[0-7]` Any of `mosfet0`, `mosfet1`, . . . , `mosfet7`

`--included-cells-file=`*name* This is like the `--excluded-cells-file` option, but EMX will only include cells that match (or subcells of those cells), instead of excluding them.

`--device-cells-file=`*name* Any cells in the GDSII file whose name matches one of the regular expressions in the specified file will be treated as "black boxes." The top-level labels (or pins, if you're using `--cadence-pins`) in these cells will become extra ports, but all other geometry in the cells will be ignored. This option is useful when simulating layouts that contain embedded devices such as transistors. See section 19 for more details.

`--no-via-align` If you have vias that overlap, EMX by default tries to shift them around a bit and adjust the sizes so that the vias exactly correspond. This simplifies the mesh, but as currently implemented the alignment process can take a long time if there are large numbers of vias (after merging). This option turns off the via alignment and can sometimes speed up the simulation significantly despite possibly increasing the number of mesh elements.

`--colinear-fuzz=`*number* EMX normally simplifies input polygons by eliminating "nearly colinear" vertices. Three consecutive vertices on the boundary of a polygon are considered nearly colinear if the segment between the first and third ones passes within a certain distance of the middle one. If the segment is close enough, then EMX eliminates the middle point. The distance for this default simplification is 0.005 microns. You can change this behavior by giving a different number of microns. If you specify 0, this simplification is turned off.

`--internal-units-per-micron=`*number* For meshing, the geometry in EMX is represented by polygons whose vertices lie on a grid. The internal grid spacing defaults to one nanometer. This option changes the number of grid units per micron. Increase it to make the grid finer; for example, setting this option to 10000 makes the internal grid spacing 0.1 nm.

`--no-adaptive-via-mesh` Normally the edge width for meshing a via depends on the edge widths of the metals that the via connects. This option makes the via edge width always equal to the `--edge-width` value.

20.2 Ports

`--include=`*port* Normally EMX solves for all the ports in the structure. You may want to solve for only some of the ports however, leaving the rest always grounded. If this is the case, use `--include` to give the names of the ports that you're interested in. You can specify this option multiple times, e.g., `--include=p1 --include=p3`. The abbreviation for `--include` is `-i`.

`--exclude=`*port* If you want to solve for almost all of the ports, you can just list the ones that you don't want to solve for using `--exclude`. Like `--include`, you can specify `--exclude` multiple times. You may abbreviate `--exclude` with `-x`. Note that, as with `--include`, the ports that are not solved for are still grounded.

`--port=`*port specifier* By default, EMX attaches a voltage source from each label in the structure to the ground plane. If you want to specify explicit ground references, or if you don't want all the default source connections, then you need to provide a series of port specifiers. If you use `--port`, then *no default voltage sources are attached to the structure.* Each port specifier has the form *port name*`=`*signal label list*`:`*ground label list.* (The `:` can be changed to another character; see `--signal-ground-separator`). The label lists are sequences of one or more labels, separated by commas. The ground label list may be omitted, in which case the port is connected to the ground plane. You can also omit the signal label list, in which case the port is attached from the label of the same name to the ground plane. Here are some examples of port specifiers:

`p1` A port named `p1` connected from label `p1` to the ground plane.

`p1=s` A port named `p1` connected between label `s` and the ground plane.

`p1=s:g` A port named `p1` connected between label `s` and label `g`.

`p1=s:g1,g2` A port named `p1` connected between label `s` and labels `g1` and `g2`.

The current for a port, which is what EMX prints when it prints the Y-parameters, is the total current flowing across all of the edges in the signal label list. For ports with explicit grounds, all of this current must come from other edges in the port, so that the total current across all of the port's edges is zero. Multiple ports may have the same ground label list; for example, in order to simulate a GSGSG probe, use:

```
... -p p1=s1:g1,g2,g3 -p p2=s2:g1,g2,g3 ...
```

When simulating `p1`, the voltage between `s1` and the grounds is 1 V, `s2` is grounded, and the current flowing into `s1` comes from `s2` and the grounds. You may abbreviate `--port` as `-p`.

`--signal-ground-separator=`*character* This sets the separator for the signal and ground parts of a port to the specified character. If some of your labels contain a colon then this option may be useful. For example,

```
... --signal-ground-separator=/ -p P1=s:/g: ...
```

can be used to declare a port containing the labels `s:` and `g:`.

`--mode=`*mode specifier* The default driving mode for a port is to stimulate the port with a 1 V source while all other ports are grounded. You can override this by specifying one or more explicit `--mode` options. A mode specifier is a sequence of port names prefixed by coefficients, as in these examples:

`+p1` The default mode for port `p1` (1 V on `p1`, all other ports grounded).

`+p1-p2` A differential mode, 1 V on `p1` and −1 V on `p2` (with all other ports grounded).

`+p1+p2` A common mode, 1 V on `p1` and 1 V on `p2`.

`-2*p1+3.5*p2+p3` −2 V on `p1` and 3.5 V on `p2`, and 1 V on `p3`.

The modes that you specify should be reasonably orthogonal, that is, there should not be a linear combination of the modes that adds up to something close to zero. EMX automatically completes the set of modes that you specify with additional orthogonal modes. In the common case where you want differential-mode and common-mode stimulation of two ports, you can just specify one mode and EMX will derive the other one. That is, these are essentially equivalent:

```
emx ... --mode=+p1-p2 ...
```

and

```
emx ... --mode=+p1+p2 ...
```

Regardless of the specified modes, EMX always prints the output parameters in the same form, corresponding to individual stimulation of each port.

`--internal=`*specifier* In order to attach a port to the interior of a conductor, you must declare the associated label as internal using this option. The specifier has the form of a label name (or a wildcard `*`), an optional conductor name, and one or two numbers, all separated by commas. For example:

`p1` Label `p1` is an internal label.

`p1,4.2` Label `p1` is an internal label with size 4.2 microns (a 4.2 by 4.2 micron square).

`p1,4,3` Label `p1` is an internal label with size 4 microns wide by 3 microns high.

`p1,metal1,4.2` Label `p1` on conductor `metal1` is internal with size 4.2 microns.

`*,2` All labels are internal with size 2 microns.

`*,metal1,3` All labels on conductor `metal1` are internal with size 3 microns.

The default shape for an internal label is a one micron square. EMX creates a rectangle of the appropriate size centered on the label. Current from the port's source will be injected into any conductors within this rectangle. If there are no conductors inside the rectangle, EMX prints an error message and exits. If a label is matched by more than one internal specifier, then the size is determined by the most specialized one. The `--internal` option can be abbreviated as `-I`. If you're using Cadence, we recommend using rectanglar shape pins for internal ports; see the `--cadence-pins` option for details.

`--label-depth=`*number* By default, EMX only considers labels in the top level GDSII structure when looking for places to attach ports. If you happen to have labels further down in the hierarchy, you can specify how deep EMX should look using this option. The default, 0, means only look at the top level. Specifying `--label-depth=1` would mean to look in sub-structures of the top level structure too. The abbreviation for `--label-depth` is `-l`.

`--cadence-pins=`*number* When a GDSII file is streamed out from Cadence, there is an option to write special property records in the file that encode pin information. This option tells EMX to interpret the pin information and add labels to the layout at the pin locations. The number is a GDSII property number; it must be specified in Cadence at stream out time, and you must specify the same number to EMX. The text of the label comes from the pin name. EMX presently only handles shape pins. The location of the label is at the centroid of the shape, except in the case of a rectangular shape pin. For rectangular pins, if the pin has a single access direction (top, bottom, left, or right), then the label is located on the corresponding edge of the rectangle. Typically you would use this for pins that should become edge ports in EMX. (Note that the label will snap to the nearest conductor edge, which is not necessarily just the edge of the pin rectangle.) If a rectangular pin has *all four* access directions, then EMX automatically treats the label as an internal port whose size matches that of the rectangle. Note that shape pins are often on special GDSII layers. For EMX to see these, you must be sure to reference the appropriate layer when defining conductors. For example:

```
define m1 = L30T0+L130T0  # L30 = drawing, L130 = pin
```

When you specify this option, then only pins are used to generate labels. All regular (non-pin) labels in the layout are ignored.

`--bottom=`*specifier* Normally internal connections use the top surface of the conductor as the connection point. You can use the bottom instead by specifying this option. The specifier consists of an optional negation flag followed by a label name (or wildcard) and an optional conductor name (or wildcard). Example specifiers:

`p1` Any internal label `p1` should attach to the conductor bottom.

`-p1` The internal label `p1` should not be on the bottom.

`*,metal7` All internal labels on `metal7` should be on the bottom.

`*` All internal labels should be on the bottom.

`-p2,metal3` The internal label `p2` on `metal3` should not be on the bottom.

You can give multiple `--bottom` arguments. For an internal label on some particular conductor, the most precise specifier that matches the label/conductor is used to determine whether it's on the bottom or top. Consider this example:

```
--bottom=* --bottom=-P1 --bottom=P1,metal3
```

Then all internal labels will be on the bottom except for those `P1` labels that are not on `metal3`.

When a conductor is declared as `internal bottom`, then the default location for internal connections is the bottom. In that case, a negated declaration like

```
--bottom=-P8,metal2
```

can be used to put a connection on the top.

`--rename-labels=`*conductor,format* This causes EMX to rename all the labels on the specified conductor according to the given format. The format should contain a `%s` which will be replaced by the label. For example

```
... --rename-labels=bga,%s_pkg ...
```

would cause a label `vss` on conductor `bga` to be renamed to `vss_pkg`.

`--unique-labels` Makes EMX add a numeric suffix to all labels so as to ensure that all labels are distinct. For example, multiple occurrences of label `a` become `a_1`, `a_2`, etc. This is mostly useful if you have multiple instances of a single cell with some labels in it and you want all of the labels to remain distinct. The numbering is done at the GDSII level and thus *before* any relabeling specified by `--rename-labels`.

`--label-snap-distance=`*number* EMX does not require the port label for an edge to be exactly on the edge. By default, the label will "snap" to the closest edge within four microns. If there's no edge within that distance, EMX prints an error. You can change the distance with this option. For labels that are terminals of black-boxed devices, there is a different distance controlled by `--device-label-snap-distance`.

`--device-label-snap-distance=`*number* Like `--label-snap-distance`, but applied to the terminals of black-boxed devices. The default distance is 0.01 microns, i.e., you have to route basically exactly to the label.

`--device-pins-internal` Cadence pins that are edge pins (i.e., that have access direction top, bottom, left, or right) normally become labels, and these then snap to nearby edges. However if the pins are within black-boxed devices, then this behavior can sometimes be inconvenient if the routing to the device is a bit sloppy. This option tells EMX to treat any Cadence shape pin in a black-boxed device as if it has access direction TBLR, so the pin will become an internal port for EMX.

`--uniform-sources` The default source formulation allows current to jump from one part of a port connection to another through a zero impedance return path. This option eliminates those paths and forces the current to flow uniformly into or out of each connection region.

20.3 Process file control

`--define=`*name,number* This is almost equivalent to adding a definition of the form:

```
define name=number
```

in the process file. The "almost" is because you cannot have two definitions for the same name in the process file, but a definition on the command line is allowed to override a default definition in the process file.

`--definitions-file=`*filename* This specifies an additional process file which will be read before EMX reads the main process file. A typical use is to put layout simplification operators in the definitions file and the layer stack-up and physical process description in the main process file. This option may be given multiple times. The additional definition files are read in order before the main process file.

`--corner=`*name* Request simulation at a particular process corner. Any random variable in the process file with a corresponding `corner` value will be fixed at that value. Multiple `--corner` options may be given, but only one may apply to any particular random variable.

`--temperature=`*number* Set the simulation temperature to the specified value. This determines the value of the special `deltat` and `dtemp` variables in the process file. See section 13.

`--width=`*number* Assume that the local geometry width is equal to the specified number of microns (instead of calculating it).

`--spacing=`*number* Assume that the local geometry spacing is equal to the specified number of microns.

`--no-geometry-dependence` Turn off local width and spacing computation. This can make the computation run faster if there are many geometry dependencies in the process file but you only want a rough answer, or you know that the dimensions in the layout are such that the geometry-dependent effects are minor.

`--case-sensitive` By default identifiers in the process file are internally converted to lower case. This option makes EMX preserve the case of identifiers instead.

`--allow-nonphysical-mu-epsilon` This keeps EMX from complaining if the relative permittivity or permeability of a layer is less than one.

`--mask=`*llx,lly,urx,ury* This is like adding an `emxmask` statement to the process file, with the mask being a rectangle defined by the specified bounding box. If there is another `emxmask` statement in the process file, then the mask is the intersection of that region and the box.

`--encrypt-process` Create a fully encrypted process file `encrypted.proc`. See section 16 for details.

`--partial-encrypt-process` Create a partially encrypted process file called `encrypted.proc`; see section 16.

`--key=`*string* Set the key for encryption and decryption of the process file. When you use `--key` without one of the process file encryption options, then the process file will be decrypted internally, i.e., EMX will behave as if you had used an unencrypted process file without `--key`. For information on encrypting process files, see section 16.

20.4 Visualization

`--geomview-mesh` This option tells EMX to compute the mesh for the structure and then write an output file `mesh.list`, which may be viewed with `geomview`. `Geomview` is freely available at `http://www.geomview.org/`. The abbreviation for `--geomview-mesh` is `-g`. EMX stops after printing the mesh, so you may omit the frequencies.

`--geomview-mesh-and-continue` This is like the `--geomview-mesh` option, but EMX continues after printing the mesh. The abbreviation is `-G`.

`--matlab-mesh` This option tells EMX to compute the mesh for the structure and then write an output file `emxmesh.m`, which may be viewed under MATLAB[†]. The abbreviation for `--matlab-mesh` is `-m`. EMX stops after printing the mesh, so you may omit the frequencies. To view the mesh, start MATLAB in the directory where `emxmesh.m` resides, and type `emxmesh`. Type `help emxmesh` to see other options.

`--matlab-mesh-and-continue` The abbreviation for this option is `-M`, and it makes EMX continue the simulation after writing the MATLAB-format mesh.

`--print-process` This option makes EMX read the process file and print an output file with a picture of the process cross section (like figure 2). For this option, only the process file is relevant, so you may omit the GDSII file, the structure name, and the frequencies. By default the output is a PostScript file `proc.ps`, but see the next item.

`--generate-pdf` Specifying this along with `--print-process` will make EMX produce a PDF file `proc.pdf` for the process cross section.

`--process-comment`**string** Typically the string would be something like the name of the process, a version number, a process corner, etc. Multiple instances of this option may be given; each string is shown on a separate line. The option `--user-comment` is a synonym.

`--rescale-dielectrics` and `--rescale-conductors` When EMX prints the process cross section the conductors, vias, and dielectric layers are shown to scale by default (with the exception of very thick dielectric layers). As a result, there is often insufficient room to show the layer properties for thin layers, or the names for thin conductors and vias. These options makes EMX stretch thin layers, conductors, and vias enough for them to be labeled.

`--print-currents` This option makes EMX write a MATLAB-format output file `emxcurrent.m` showing the horizontal current density in the structure. The plot shows the ratio in dB of the horizontal current density

[†]MATLAB is a registered trademark of The Mathworks, Inc.

to the maximum horizontal current density. Horizontal current density is calculated by integrating the current density over the vertical thickness of the conductors. So if two wires have the same current density and one is twice as thick as the other, then the plot will show a difference of 6 dB. To view the plot, start MATLAB in the directory where `emxcurrent.m` resides, and type `emxcurrent`. Type `help emxcurrent` to see other options.

`--print-charges` This option makes EMX write a MATLAB-format output file `emxcharges.m` showing the charge density in the structure. The plot shows the ratio in dB of the charge density to the maximum charge density. To view the plot, start MATLAB in the directory where `emxcharges.m` resides, and type `emxcharges`. Type `help emxcharges` to see other options.

20.5 Output

`--y-file=`*filename* If you are simulating many frequency points, you probably want to save the results to a file in order to plot them with something like MATLAB. The `--y-file` option tells EMX to write the Y-parameters to the specified file. This option may be abbreviated as `-y`. The format of the data is specified with a *preceding* `--format` option. You can specify `--format` and `--y-file` more than once to get multiple output files in different formats.

`--s-file=`*filename* The `--s-file` option is similar to `--y-file`, but it makes EMX write S-parameters to the specified file instead of Y-parameters. The abbreviation for this option is `-s`. The format of the data is specified with a *preceding* `--format` option. The reference impedance is given by a *preceding* `--s-impedance` option.

`--format=`*type* This option specifies the output format for Y- and S-parameter files. The valid choices are:

`matlab` For an n-port structure, each line has $2n^2 + 1$ floating-point numbers, separated by spaces. The first number is the analysis frequency. The remaining numbers are real(A_{11}), imag(A_{11}), real(A_{21}), imag(A_{21}), ..., real(A_{nn}), imag(A_{nn}), where A is the response matrix (either Y- or S-parameters).

`append-matlab` The format is like `matlab`, but EMX appends the results to the file instead of overwriting the file. This is the default format.

`touchstone` The file will be in TOUCHSTONE format[‡] If the file name for a TOUCHSTONE file ends with `.s%dp` or `.y%dp`, the `%d` is replaced by the number of ports being simulated.

`spectre` The file will be in the SPECTRE[§] circuit simulator n-port format.

[‡]TOUCHSTONE is a registered trademark of Agilent Corporation.
[§]SPECTRE is a registered trademark of Cadence Corporation.

`psf` The output will be in Cadence PSF format. In this format, the output filename specifies a directory. The actual data file within this directory has a standardized name (for example, `Y.ac` for Y-parameters). If the directory does not exist, it will be created. The PSF-format outputs can be viewed with the standard Cadence plotting tools.

You may abbreviate `--format` with `-f`. The format names may also be abbreviated: `-f touch` is the same as `-f touchstone`. You can give `--format` and `--y-file` or `--s-file` multiple times to get multiple files in different formats. For example, to get output file `ind.s2p` in Touchstone format and file `ind.data` in Spectre format, use

```
... -f touchstone -s ind.s2p -f spectre -s ind.data ...
```

`--format` is also used for giving the output format for model files (see subsection 20.9).

`--s-impedance=`*number* This option sets the port reference impedance when printing S-parameters. The default is 50 ohms.

`--touchstone` This is the same as `--format=touchstone`.

`--include-command-line` With this option, EMX will print the command line as a comment along with the simulation data. Only Touchstone and Spectre formats support this.

`--include-port-order` This makes EMX print the port order as a comment in the simulation data, again only in Touchstone and Spectre formats.

`--print-command-line` This option makes EMX print the command line as part of the output. Mostly this is useful if you're capturing a log file and want to see what options were used for a particular EMX run.

`--user-comment=`*string* Sets a comment that will be included in the output. The option is synonymous with `--process-comment`, and hence the comment is also shown when doing `--print-process`.

`--print-double` This makes EMX print the data in the output files in double precision. By default, EMX prints single-precision results.

20.6 Logging

`--verbose=`*number* This option controls the amount of information printed by EMX. The default is `--verbose=1`, which causes EMX to print the Y-parameters. If you are saving the Y-parameters to a file with `--y-file`, then you may want to turn off all printing with `--verbose=0`. Numbers greater than 1 cause EMX to print various statistics while running.

`--log-file=`*filename* Print status information to the specified file in addition to the terminal.

`--log-level=`*number* By default, the amount of information printed to the log file corresponds to what you get from `--verbose=3`. If you want less information, you can specify a lower number with this option.

20.7 Frequencies

`--sweep` This option makes EMX perform an adaptive frequency sweep and print the frequency response for the structure at a very fine frequency spacing. EMX will perform a sweep between the minimum and maximum frequencies that you specify. If you specify only one frequency, that is taken to be the maximum, and the minimum is assumed to be DC. If you specify more than two frequencies, EMX will do explicit solves at those frequencies, and will print results at those frequencies (in addition to the normal sweep frequencies). Because of the volume of output, EMX does not print the results on the console. As a result, you should also specify a file for saving the Y-parameters or S-parameters using the `--y-file` or `--s-file` options. So the command line for using `--sweep` should be something like this:

```
emx ... --sweep 1e8 10e9 --touchstone --s-file=out.s2p
```

`--sweep-num-steps=`*number* When using the `--sweep` option, EMX by default plots 100 points within the frequency range. You can specify a different number of points using the `--sweep-num-steps` option.

`--sweep-stepsize=`*number* If you want a frequency sweep with a certain fixed spacing between the points, use the `--sweep-stepsize` option and specify the desired spacing. For example, `--sweep-stepsize=1e8` would request points every 100 MHz.

`--multi-sweep=`*specifier* This is like `--sweep`, but you can give it multiple times if you want different step sizes when printing different frequency ranges. The specifier has one of three forms:

`1e8-10e9` is like `--sweep 1e8 10e9 --sweep-num-steps=100`

`1e8-50-10e9` is like `--sweep 1e8 10e9 --sweep-num-steps=50`

`1e8:1e7:10e9` is like `--sweep 1e8 10e9 --sweep-stepsize=1e7`

`--discrete-frequency=`*number* When used with `--sweep` or `--multi-sweep`, this option requests a separate simulation at a specific frequency, which is outside the sweep range. You can specify `--discrete-frequency` more than once; it is often used to do a few extra low frequencies. Any discrete frequencies are done and printed *before* the sweep results.

20.8 Statistical simulation

`--monte-carlo=`*number* Run the specified number of Monte Carlo simulations. The process file should contain a specification of the statistical variations in the process, as discussed in section 10. You will want to save the results of each simulation to a separate file. To do this, include a `%d` somewhere in the file name. The `%d` will be replaced by the Monte Carlo simulation number. For example

```
... --monte-carlo=20 -s ind.%d.s2p ...
```

will produce output files `ind.1.s2p` through `ind.20.s2p`.

`--random-seed=`*number* Do a simulation with random variations. If the specified number is *n*, then the simulation uses the same process parameters as the *n*th simulation in a Monte Carlo analysis. You might want to use this option with `--print-process` to view the process parameters chosen for a particular Monte Carlo simulation. The `random-seed` option may also be used to restart a Monte Carlo simulation. If you previously ran `--monte-carlo=100`, then running:

```
... --monte-carlo=100 --random-seed=101 ...
```

will do 100 additional simulations. The effect is the same as if you had specified `--monte-carlo=200` initially.

`--perturbation` Run a perturbation analysis in addition to the normal simulation. For each output file, there will be a series of auxiliary output files, one per independent random variable in the process file. If an output file is called, e.g., `ind.s2p`, then the auxiliary files are by default named `d1_ind.s2p`, `d2_ind.s2p`, etc. The *n*th output file contains derivatives with respect to the *n*th random variable, scaled by the standard deviation of the variable. The auxiliary output files have the same format as the main file. You can change the names of the auxiliary files with `--derivative-file`.

`--derivative-file=`*name* This options sets the name of the derivative files corresponding to the most recently specified output file. The name should contain either `%d` or `%s`. A `%d` will be replaced by the number of the random variable. For example

```
... --s-file=ind.s2p --derivative-file=ind_d%d.s2p ...
```

produces derivative files `ind_d1.s2p`, `ind_d2.s2p`, etc. A `%s` will be replaced by the name of the random variable.

20.9 Model creation

`--model-file=`*name* This option tells EMX to create a pole-zero (state-space) model and save it in the given file. You can give this option multiple times to save the model in different formats.

`--format=`*type* The `--format` option specifies the syntax used for the model. Valid types are:

- `spice` The output will be a Spice subcircuit. This is the default format. This format should also work for Synposys's HSPICE[¶]
- `spectre` This type requests a subcircuit for Cadence's SPECTRE simulator.
- `cktrom` This produces a subcircuit containing a `cktrom` element for SPECTRE. (This is often a bit faster than the regular form with controlled sources for models of small-to-moderate size, but may become slower as the model size gets larger.)

In all cases, the name of the subcircuit is the same as the name of the model file, but without an extension. If you specify a file name with no extension, an appropriate extension (`.sp` for Spice and `.scs` for SPECTRE) is automatically added. You should specify the `--format` *before* the corresponding `--model-file` option.

`--include-command-line` Include the EMX command line as a comment in the output model file.

`--include-port-order` Include the EMX port names as a comment.

`--model-tolerance=`*number* This specifies how strict EMX should be in trying to match the model to the frequency-domain simulation. The default tolerance is 3×10^{-4}, but you can try loosening it if the model creation is taking too long or if the model is too big. You can also tighten it if you want the model to fit more closely. It is not guaranteed that EMX will be able to meet the requested tolerance.

`--model-max-poles=`*number* Limits the number of poles to at most the specified amount. EMX will still adaptively try to produce a model with fewer poles that meets the model tolerance. Also, there is a minimum number of poles that the fitting procedure starts with; if you specify a limit of less than this, the minimum number is used.

`--model-num-poles=`*number* Give this option to request a model with the specified number of poles. (The actual number of poles will be at least the requested amount, but may also be slightly larger.) With this option, EMX ignores the model tolerance (and also any `--model-max-poles` option).

`--model-max-time=`*number* This limits the wall-clock time for the model construction to at most the specified number of seconds. If the model tolerance is met before the time limit, the construction will stop earlier.

[¶]HSPICE is a registered trademark of Synopsys, Inc.

`--model-s-impedance=`*number* Makes EMX use the given reference impedance when computing the S-parameters that the model should match. By default, reference impedances are computed to try to minimize the maximum of all $|S_{ij}|$. You can also override individual reference impedances using `--model-s-impedance=`*port,number*. If any impedance is specified, then the unspecified impedances default to 50 ohms.

`--model-per-frequency` Compute S-parameter reference impedances for each frequency separately. Normally one set of reference impedances is used for all frequencies.

`--save-model-state` This option makes EMX dump out the information for creating a model into a file called `emxmodinfo.dat`. We recommend that you use this option when creating models so that if the model isn't satisfactory, you can rerun the model creation without having to redo the simulation from scratch. You can also give this option even if you don't request model creation; doing so will allow you to create a model later. See the next option for details.

`--load-model-state` Specifying `--load-model-state` tells EMX to read the `emxmodinfo.dat` file and run only model generation. You can specify different modeling options, output files, or output formats than you did when running with `--save-model-state`. Other simulation options are ignored when you use this option.

`--model-reduce-only` If the dielectrics in the process are all lossless (or at least very nearly so) and all the material properties are frequency-independent, then you can use this option to invoke an alternative method of model creation which is much faster. Even when the method is applicable, it may produce slightly larger models than the default algorithm. If the dielectrics have too much loss, this method will produce a poor fit.

20.10 Types of simulation

`--capacitive-only` This tells EMX to solve only the capacitive part of the problem. The answer will have the form of a capacitance matrix with entries $g + j\omega c$. The loss part, g, will be approximately zero unless some of the layers have a nonzero conductivity or loss tangent.

`--inductive-only` This option makes EMX solve only the inductive (and resistive) part of the problem. This is essentially a magnetostatic solve.

`--resistive-only` Make EMX solve only the resistive part of the problem. The analysis frequency is irrelevant with this option.

`--RC-only` This option makes EMX solve only the resistive and capacitive parts of the problem. This can save some time if the inductive effects are known to be mostly irrelevant.

`--frequency-independent` This tells EMX to use only the DC Green's functions even if the process involves frequency-dependent quantities (either explicitly, or in the form of layer conductivity). It makes EMX's output more like that of a static layout extractor. The advantage is that the simulation is faster and requires less memory, since the Green's function doesn't have to be tabulated at multiple frequencies. Also, if you're making pole-zero models and don't have loss tangents in your process, then you can use the `--model-reduce-only` option, which can be much faster if you've got lots of ports. The disadvantage is that you lose some accuracy. For IC processes, the biggest effects are usually an increase in the substrate capacitance and a decrease in the substrate loss. This option is probably most useful in cases where you're making models of smaller subcells using the black-box flow.

`--quasistatic` EMX calculates the maximum possible retardation effects given the size of the structure and the highest simulation frequency. If this conservative estimate shows that the effects must very small, then EMX uses a quasistatic model when calculating the interactions between elements. Otherwise, EMX uses a full-wave model. The quasistatic calculation is faster than the full-wave one. In some cases, you may know beforehand that the retardation effects are not interesting. For example, the well-separated parts of the structure may not couple strongly, or the maximum frequency might correspond to a second or third harmonic where less accuracy is required. In these cases, you can force EMX to use the quasistatic model with the `--quasistatic` option. (Alternatively, you can force EMX to use the full-wave model with `--full-wave`.) If you are unsure of whether retardation is significant, you may want to run the highest simulation frequency in both modes before running an adaptive frequency sweep or a large number of simulation points.

`--full-wave` This forces EMX to use full-wave interaction computations. See the discussion under `--quasistatic`.

`--dump-connectivity` Tells EMX to produce a file `emxdc.dat` containing the DC conductance matrix obtained from the resistive part of the problem.

`--print-connectivity` This makes EMX print the matrix of port-to-port DC resistances. Each value is the result of open-circuiting all ports except for one pair and getting the equivalent resistance seen by a one volt source between the two ports. In the printed matrix, `0` means less than 10^{-9} ohms and `-` means no connection (or more than 10^{12} ohms). The diagonal entries correspond to resistance to a grounded port.

`--solve-for-s` Normally EMX solves internally for Y-parameters (though it can also output S-parameters). This option makes EMX solve internally for S-parameters. This can improve numeric stability in rare cases that involve extremely low impedances (for example, pairs of ports on either side of a short, wide strip of metal).

20.11 Efficiency and accuracy

`--selected-sweep` When EMX is doing a frequency sweep, by default it solves for each port at each frequency. This option makes EMX check to see whether the solution for each port is "stable" within the current frequency range. "Stable" means that the current flow when the port is stimulated is approximately the same throughout the whole range. When EMX starts simulating at a new frequency, it only solves for ports that are not yet stable. This option can greatly reduce simulation time if there are many ports, and is generally recommended. It is the default if there are at least ten ports.

`--no-selected-sweep` This cancels the implicit `--selected-sweep` if there are ten or more ports.

`--relative-tolerance=`*number* EMX solves the discretized system to a specific tolerance. Small entries in the Y-parameter matrix that are on the order of this tolerance times the largest values in their column can be very inaccurate. One option that controls the solve is `--relative-tolerance`. This specifies the reduction in the norm of the residual. If the discretized system is $Ax = b$, then the relative norm of the residual for an approximate answer $\tilde{x}$ is $|A\tilde{x} - b|/|b|$. The default relative tolerance is 10^{-5}.

`--answer-relative-tolerance=`*number* This is the second option controlling the relative tolerance of the answer. The answer printed by EMX is $P\tilde{x}$, where $\tilde{x}$ is the approximate answer and P is a matrix that adds up the currents flowing in each port. Generally the answer will converge faster than the overall relative norm of the residual. EMX monitors $P\tilde{x}$ during the solve and stops if this answer stops changing, even if the relative tolerance has not been reached. The relative tolerance for the change in $P\tilde{x}$ from one step to the next is set by `--answer-relative-tolerance`, which defaults to 10^{-3}. You may turn off the possible early termination of the solve by setting the answer relative tolerance to zero. If you just want more precision, set the answer relative tolerance to 10^{-4}.

`--double-precision` In some situations, such as with capacitors which have a relatively high Q, EMX may give noisy results even with tighter solve tolerances. This noise can often be cleaned up by giving the option `--double-precision`. This doubles the memory required for the solve though.

`--dc-relative-tolerance=`*number* Like `--relative-tolerance` except it applies only to a solve at DC. The default is 10^{-7}.

`--dc-answer-relative-tolerance=`*number* This option defaults to 10^{-5} and is like the `--answer-relative-tolerance` option but only for a solve at DC.

`--max-blocking=`*number* By default, EMX solves for multiple ports at the same time. For large simulations, this may use too much memory. You can limit the number of ports that are solved for simultaneously by setting `--maximum-blocking`. The default is 10, i.e., EMX will solve for up to 10 ports at once. We do not recommend increasing this number for examples with very many ports; it is better to use `--parallel` along with a high `--max-memory` limit. This will allow EMX to do multiple independent blocks in parallel. (This option used to be called `--maximum-blocking`.)

`--cache-file=`*name* This makes EMX save calculated any Green's function information to a file. If you use the same file for a subsequent run, some Green's function information will not need to be recomputed. If you have many layers and your structure is large enough to warrant using `--full-wave`, this option may save a small amount time.

20.12 Parallelism

`--parallel=`*number* This option makes EMX use up to the specified number of parallel threads when simulating. It is OK (and generally even beneficial) to set this number to slightly more than the number of CPUs because the load balancing is not perfect. The default is to use a single thread. If you specify `--parallel=0`, then EMX will use 1.5 times as many threads as there are physical CPUs. Using this option can affect the order in which solves are done, which may change the results slightly.

`--simultaneous-frequencies=`*number* This option allows EMX to solve for up to the specified number of frequencies in parallel. By default, the number of simultaneous frequencies is limited to one. If you set this to 0 (recommended), then EMX will do as many simultaneous frequencies as it profitably can, subject to processor and memory availability. This option can also affect the results slightly by changing the order in which solves are done.

`--max-memory=`*amount* This option sets a memory limit for EMX. The limit is not a hard bound; EMX will use more memory if it needs to. When running with multiple threads (and possibly multiple simultaneous frequencies), EMX will try to make more parallelism available by increasing memory use (up to the given limit). Low memory limits may restrict the amount of parallelism available, increasing simulation time. The limit amount can be given in several forms:

`3G` Use at most 3 gigabytes of memory.

`700M` Use at most 700 megabytes of memory.

`-1.5G` Use all the available memory on the machine except for 1.5 gigabytes.

`75%` Use 75% of the available memory.

`-10%` Use all but 10% of the available memory.

`15` Same as `15%`.

By default, EMX will use up to the larger of: 80% of the total memory, or the total memory minus 8 gigabytes.

20.13 Other options

`--version` Print EMX's version number and exit.

`--license` Print the licensing information for EMX and exit.

`--supports-feature=`*name* Exits with status 0 if EMX supports the named feature, and with status 2 if it does not. EMX versions that do not have the `--supports-feature` option at all will exit with status 1 when given this option. This option is used by the Cadence interface to EMX for auto-configuration.

`--flexnet-queue` When using FlexNet licensing, this option makes EMX queue until a license is available. By default, EMX exits immediately if all licenses are in use.

`--list-colors` Print the names that EMX recognizes for the `color` attribute of conductors and vias, and then exit.

21 Acknowledgements

EMX is based on various numeric libraries and other software. Licensing information and acknowledgements are as follows.

EMX uses LAPACK and BLAS linear algebra libraries.

Copyright © 1992–2010 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.

Copyright © 2000–2010 The University of California Berkeley. All rights reserved.

Copyright © 2006–2010 The University of Colorado Denver. All rights reserved.

Additional copyrights may follow.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Sparse matrix ordering is based on AMD and METIS.

AMD Version 1.0 (Apr. 30, 2003), Copyright (c) 2003 by Timothy A. Davis, Patrick R. Amestoy, and Iain S. Duff. All Rights Reserved.

AMD License:

Your use or distribution of AMD or any modified version of AMD implies that you agree to this License.

This material is provided as is, with absolutely no warranty expressed or implied. Any use is at your own risk.

Permission is hereby granted to use or copy this program, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses AMD or any modified version of AMD code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the

Availability note are retained, and a notice that the code was modified is included. This software was developed with support from the National Science Foundation, and is provided to you free of charge.

Availability:

http://www.cise.ufl.edu/research/sparse/amd

METIS version 5.1.0 License:

Copyright 1995-2013, Regents of the University of Minnesota

Licensed under the Apache License, Version 2.0 (the “License”), obtainable from http://www.apache.org/licenses/LICENSE-2.0; you may not use this file except in compliance with the License.

1. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
2. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
3. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   You must give any other recipients of the Work or Derivative Works a copy of this License; and You must cause any modified files to carry prominent notices stating that You changed the files; and You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the

Derivative Works; and If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

4. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
5. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
6. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, without WARRANTIES or CONDITIONS of ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under

this License.

7. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
8. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

Availability:

http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

Non-symmetric sparse matrix factorization is based on UMFPACK

UMFPACK Version 4.1 (Apr. 30, 2003), Copyright (c) 2003 by Timothy A. Davis. All Rights Reserved.

UMFPACK License:

Your use or distribution of UMFPACK or any modified version of UMFPACK implies that you agree to this License.

This material is provided as is, with absolutely no warranty expressed or implied. any use is at your own risk.

Permission is hereby granted to use or copy this program, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses UMFPACK or any modified version of UMFPACK code must cite the Copyright, this License, the Availability note, and “Used by permission.” Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. This software was developed with support from the National Science Foundation, and is provided to you free of charge.

Availability:

http://www.cise.ufl.edu/research/sparse/umfpack

Robust geometric predicates are by Jonathan R. Shewchuk.

Routines for Arbitrary Precision Floating-point Arithmetic and Fast Robust Geometric Predicates
(predicates.c)

May 18, 1996

Placed in the public domain by
Jonathan Richard Shewchuk
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15213-3891
jrs@cs.cmu.edu

MD5 code is from Alladin Enterprises.

Copyright (C) 1999, 2002 Aladdin Enterprises. All rights reserved.

This software is provided "as-is", without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

L. Peter Deutsch
ghost@aladdin.com

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. Source code for the PCRE library may be obtained from:

ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/

The PCRE library is available under a BSD license:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

PDF generation code is via the HARU library.

Copyright (C) 1999–2006 Takeshi Kanno

This software is provided 'as-is', without any express or implied warranty.

In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Compression support in PDFs is via zlib.

Copyright (C) 1995–2012 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.