EMX Virtuoso[*] Interface Reference

Version 03-Mar-2015
Integrand Software, Inc.

This manual describes the details of the Cadence Virtuoso interface for EMX.[†] For a tutorial introduction showing the typical flow for using the interface, see the associated *EMX Virtuoso Interface Tutorial*.

1 Installing the Interface

To install the interface:

1. unpack the interface and put it in an appropriate spot;
2. customize the configuration file `emxconfig.il` file for your environment; and
3. set your `.cdsinit` to load `emxconfig.il` when starting Virtuoso.

The interface is distributed as a tarball; unpack it using the command:

```
tar xzvf emxinterface.tar.gz
```

(Replace `emxinterface.tar.gz` by the appropriate file name.) This produces an `emxinterface` subdirectory with `cadence5` and `cadence6` subdirectories. You should use the interface in the `cadence5` subdirectory with Cadence version 5.xx, and the `cadence6` subdirectory for Cadence versions 6.xx and later. In this manual, we'll assume the use of Cadence version 6.xx. If you use Cadence version 5.xx, simply replace `cadence6` in the paths with `cadence5`.

Figures 1 through 4 shows the contents of the `emxconfig.il` configuration file. It contains definitions of some SKILL variables that configure the interface. The most important variable is the first one, `EMX_interface_path`, which specifies where the files for the interface have been installed. If you unpacked the interface in a place like `/usr/local`, then the `emxconfig.il` file will be located in the subdirectory:

```
/usr/local/emxinterface/cadence5/emxinterface/emxskill
```

[*]Virtuoso is a registered trademark of Cadence Design Systems, Inc.
[†]EMX is a registered trademark of Integrand Software, Inc.

Edit `emxconfig.il` and set `EMX_interface_path` to the same path but *without the final* `emxskill`:

```
/usr/local/emxinterface/cadence5/emxinterface
```

These are the other major variables (which you may not need to be customize):

`EMX_layer_table`* If your Cadence technology library does not contain GDSII layer information for stream out, then you need to supply a separate layer map file that gives the stream layers corresponding to the layers in Cadence. Set this variable to the path for that layer map file.

`EMX_process_path`* This variable gives a directory that contains the EMX process file corresponding to the technology that you're using. The default is to use the `process` subdirectory of the `EMX_interface_path` directory.

`EMX_process_name`* Specifies the file name (without a path) of the EMX process file. If you set this to `nil`, then the interface will use the first `.proc` file within the directory given by `EMX_process_path`.

`EMX_remote_machine` Normally EMX is run on your local machine, but if you want to run it on a separate server, you can set `EMX_remote_machine` to the server name. For this to work, you must be able to execute a command on the server using `ssh` without having to specify a password. Read the `ssh` documentation for information about how to set this up using public/private keys. Alternatively, if your organization uses the Platform LSF load sharing facility, you can set the `EMX_remote_machine` variable to a string of the form `bsub ...` (or `qsub ...`). The remainder of the string will be used as default options for the `bsub` command. For more details and configuration information, see subsection 4.9.

`EMX_path` This variable specifies the directory where EMX has been installed; the directory contains the executables `emx`, `modelgen`, and `gdsview`.

`EMX_modelgen_path` If this variable is set, then it gives a separate directory where `modelgen` resides. The `emx` and `gdsview` executables must still be in the directory specified by `EMX_path`.

`EMX_ps_viewer` This specifies the name of an executable for viewing a PostScript file. The most commonly installed one seems to be `ggv`, but other possibilities include `gv`, `gsview`, and `ghostview`. This is only needed for viewing the process cross-section. Newer versions of EMX can also produce PDF files instead of PostScript.

`EMX_pdf_viewer` Like `EMX_ps_viewer` but for PDF files. By default, this is the same as `EMX_ps_viewer`. This is not used unless you have a newer version of EMX (2.13 or later) and also set

```
EMX_xsection_opts="--generate-pdf"
```

```
; EMX_interface_path tells where you've installed the interface.  This
; file (emxconfig.il) is in EMX_interface_path/emxskill/
EMX_interface_path="/home/long/emxinterface/cadence5/emxinterface"

; Variables marked with *TF* can depend on the technology library.  To
; use this feature, make the variable an association list like
; (("techlib1" "value1") ("techlib2" "value2") ...).  If the tech
; library is "techlib1", the variable will have value "value1", etc.
; Use t for the tech library if you want a default value.  Example
;EMX_process_name='(("tsmcN28" "emx28.proc") ("tsmcN20" "emx20.proc") (t nil))

; EMX_layer_table (*TF*) has the layer table for streaming out GDSII
; (needed if the technology file does not have this information)
; EMX_layer_table=(strcat EMX_interface_path "/emxskill/layers.txt")
EMX_layer_table=""

; EMX_EMX_opts (*TF*) sets the default command-line options for EMX on
; the advanced options form. EMX_GDSview_opts (*TF*) is for GDSview;
; it defaults to the same value as EMX_EMX_opts.
EMX_EMX_opts=""
;EMX_GDSview_opts=""

; EMX_process_path (*TF*) tells where you keep EMX process files
EMX_process_path=(strcat EMX_interface_path "/processes")

; EMX_process_name (*TF*) is the process file name (sans path).  Nil
; means take the first .proc file in the EMX_process_path directory
EMX_process_name=nil

; If you want to run EMX on a remote machine, set this to the machine
; name (or IP address).  You must be able to execute a command on the
; remote machine via "ssh machine command" without requiring a
; password.  See the ssh man page for authentication options.  If you
; want to run EMX locally, set this to the empty string ("").  If your
; organization uses the bsub facility, you can set this to a string of
; the form "bsub ..." to invoke EMX via bsub.  Specify default options
; for things like the queue as part of the "...".  (You can also use
; qsub in the same way as bsub.)
;EMX_remote_machine="bsub -q desired_queue"
;EMX_remote_machine="qsub -q desired_queue"
EMX_remote_machine=""

; EMX_path tells where you've installed the EMX executables (emx,
; modelgen, gdsview)
EMX_path=(strcat (getShellEnvVar "HOME") "/bin")

; EMX_ps_viewer and EMX_pdf_viewer are the applications used for
; viewing PostScript or PDF files (EMX versions prior to 2.13 can only
; produce PostScript files).  Common alternatives include gv, ggv,
; evince, okular, konqueror, and acroread
```

Figure 1: Interface configuration file, part 1

```
EMX_ps_viewer="ggv"
;EMX_pdf_viewer="okular"  ; when unset, defaults to EMX_ps_viewer

; Setting EMX_log_displayer to a string makes the interface spawn an
; external process to display the log file.  The string should contain
; a %s where the name of the log file should go.  Nil means use the
; internal Cadence viewer.
EMX_log_displayer=nil
;EMX_log_displayer="xterm -title 'EMX output' -e tail -f %s"

; By default, the interface puts all of the EMX files associated with
; a particular library and cell in the subdirectory lib_cell.work of
; the current directory, where lib and cell are the Cadence library
; and cell names.  If you want to put all of these lib_cell.work
; directories in a different place, set EMX_working_dir to that place.
EMX_working_dir="./EMX_work"
;EMX_working_dir="."
;EMX_working_dir="/where/I/want/the/interface/to/put/EMX/files"

; This option is a command for starting a documentation viewer.  When
; nil, the documentation button is not shown.
;EMX_documentation=nil  ; default if variable is unset
;EMX_documentation=(sprintf nil "dolphin %s > /dev/null 2>&1 < /dev/null"
;                           (sprintf nil "%s/EMX_documentation" EMX_interface_path))
EMX_documentation=(sprintf nil "nautilus %s > /dev/null 2>&1 < /dev/null"
                          (sprintf nil "%s/EMX_documentation" EMX_interface_path))

; Set view_name to a string like "sparam" to make a new view that
; holds the S-parameters for n-port models.  If there's a shell
; variable that holds a prefix of the library directory and you want
; the n-port to be relative to that directory, then set view_prefix to
; the name of the shell variable.
;EMX_nport_sparam_view_name=nil   ; view name, e.g., "sparam"
;EMX_nport_sparam_view_prefix=nil ; name of shell var, e.g., "PROJ_ROOT"

; Variables marked as *AC* are set automatically according to the
; installed version of EMX and ModelGen.  Use EMX_auto_config=nil to
; disable this.
EMX_auto_config=t

; EMX_pole_zero_models (*AC*) enables general state-space model
; creation.  EMX must support the --model-file option.
EMX_pole_zero_models=t

; EMX_black_box (*AC*) enables black box support in the interface.  It
; requires a version of EMX that supports --device-cells-file.
EMX_black_box=nil

; EMX_connectivity (*AC*) enables DC connectivity (LVS) support.  EMX
; must support the --dump-connectivity option.
```

Figure 2: Interface configuration file, part 2

```
EMX_connectivity=nil

; EMX_frequency_indep (*AC*) enables frequency-independent Green's
; function support.  EMX must support --frequency-independent.
EMX_frequency_indep=nil

; EMX_cadence_dev_info (*AC*) if EMX and GDSview recognize Cadence
; information for black-boxed devices
EMX_cadence_dev_info=nil

; EMX_mg_global_interp (*AC*) if ModelGen recognizes the
; --global-interp option
EMX_mg_global_interp=nil

; These are the original style extracted view names (and defaults)
;EMX_symbol_name="symbol"
;EMX_symbol_nport_name="symbol_nport"
;EMX_schematic_name="schematic"
;EMX_schematic_nport_name="schematic_nport"
;EMX_schematic_pz_name="schematic"
;EMX_schematic_dc_name="EMX_dc"
; These are the new style extracted view names
;EMX_symbol_name="symbol"
;EMX_symbol_nport_name="symbol"
;EMX_schematic_name="EMX_model"
;EMX_schematic_nport_name="EMX_spar"
;EMX_schematic_pz_name="EMX_pz"
;EMX_schematic_dc_name="EMX_dc"
; Optional views produced with symbol and symbol_nport
;EMX_aucdl_name="auCdl"  ; string, or nil to not generate
;EMX_aulvs_name="auLvs"  ; string, or nil to not generate
; Miscellaneous
;EMX_bsub_default=nil  ; t => default "Use bsub" field to selected
;EMX_viewproc_local=t  ; nil => emx --print-proc via bsub instead of locally
;EMX_spectre_local=t   ; nil => run spectre via bsub instead of locally
;EMX_modelgen_local=t  ; nil => to run modelgen via bsub instead of locally
;EMX_spectre_command="spectre"  ; maybe "PATH=/path/to/cadence/bin: spectre"
;EMX_modelgen_path=EMX_path  ; if modelgen is in a different place
;EMX_internal_PSF_reader=nil  ; probably not needed anymore
;EMX_xsection_opts=""  ; use --generate-pdf for PDF instead of PostScript
;EMX_specialized_models=t  ; nil => modelgen models use generic model form
;EMX_RLCK_model_schematics=t  ; nil => non-generic modelgen models have RLCKs
      ; instead of an instance reference
;EMX_new_style_menu_trigger=t  ; probably not needed anymore
;EMX_fixed_cell_name=t ; nil => use [@cellName] in black box symbol
;EMX_dialog_msgs=0 ; 0 => warn and info use dialog box
   ; 1 => warn uses dbox; info goes to CIW
   ; 2 => warn and info go to CIW
;EMX_form_init_proc=nil ; symbol = function to call after form creation
  ; arguments are basic form and advanced form
```

Figure 3: Interface configuration file, part 3

```
;EMX_schematic_creation_hook=nil ; symbol or function = hook to
 ; call after schematic view creation
 ; arguments are (open) cellview; names
 ; of lib, cell, and view; and either
 ; string (= model type) or symbol
 ; (= 'nport for S-param view, or
 ; 'generic for other schematic)

;; To make the interface set temperature coefficients for built-in
;; models:
; EMX_default_tcr1="1e-3" ; Set EMX_default_tcr1 and _tcr2 as desired
; EMX_default_tcr2="2e-3"
; EMX_schematic_creation_hook='EMX_set_model_temp_coeffs

; Now actually load the interface
(load (strcat EMX_interface_path "/emxskill/emxform.ils"))
```

Figure 4: Interface configuration file, part 4

`EMX_working_dir` The interface creates temporary files in the directory specified by this variable. The temporary files are placed in subdirectories of `EMX_working_dir` that have names like `libraryname_cellname.work`. Common choices for this variable include either the current directory `.` (where Virtuoso is running), a subdirectory of that directory (such as `./EMX_work`), or a fixed directory somewhere else.

`EMX_documentation` You can set this to a string containing a command for displaying some documentation (typically this document, or the related tutorial document).

`EMX_new_style_menu_trigger` This setting changes the method used for installing the EMX menu within Cadence. If you run into problems with some menus not showing up, try setting this to `nil`.

`EMX_EMX_opts`* Set this string to any appropriate default EMX command-line options.

`EMX_GDSview_opts`* Like `EMX_EMX_opts`, but for GDSview. If unset, this variable takes the same value as `EMX_EMX_opts`.

Variables that are marked with a * are special in that they're allowed to depend on the technology library. To use this, set the variables to association lists, keyed by the name of the Cadence technology libraries. For example, suppose that you're using two processes, 20 nm and 28 nm, and that 28 nm is a scaled version of a 32 nm process. Assume that the Cadence technology libraries are called `cds20` and `cds28`; that you want to put all the process files in one central directory; and that the EMX process files for the two technologies are `emx20.proc` and `emx28.proc`. You could set up the variables in `emxconfig.il` as follows:

```
EMX_process_path="/path/to/process/files"
EMX_process_name='(("cds20" "emx20.proc") ("cds28" "emx28.proc"))
```

```
EMX_EMX_opts='(("cds20" "") ("cds28" "--scaling=0.9"))
EMX_GDSview_opts=EMX_EMX_opts
```

If you use `t` as the last key in an association list, the associated value serves as a default.

The following variables enable various features of the interface that are available with more recent versions of EMX. They require that the installed versions of EMX and GDSview support appropriate command-line options.

`EMX_auto_config` The default is `t`, enabling auto-configuration of the remaining options.

`EMX_pole_zero_models`† Enables pole-zero model support. See subsection 3.11.

`EMX_black_box`† Enables black-boxed cell support. See subsection 3.13.

`EMX_connectivity`† Enables support for DC view generation (useful with black boxes).

`EMX_frequency_indep`† Enables frequency-independent Green's function support.

`EMX_cadence_dev_info`† Enables support of black-box cell boundaries.

The variables that are marked by † will be set automatically according to the version of EMX that is installed provided that you are using EMX version 4.3 or later. If you need to set the variables by hand (e.g., because you are using an older version of EMX or because EMX is only installed on a remote machine), then disable automatic configuration by setting `EMX_auto_config` to `nil` and then setting the remaining variables to `t` or `nil` as desired.

There are five variables that control the view names created by interface when you make symbol or extracted views. These are set by default for compatibility with older versions of the, but we recommend setting them as suggested here for more flexibility.

`EMX_symbol_name` the symbol view when the device type is not `N-port`; default (and recommended) `symbol`

`EMX_symbol_nport_name` the symbol view when the device type is `N-port`; default `symbol_nport`, recommended `symbol`

`EMX_schematic_name` the extracted lumped model view; default `schematic`, recommended `EMX_model`

`EMX_schematic_pz_name` the extracted pole-zero view; default `schematic`, recommended `EMX_pz`

`EMX_schematic_nport_name` the S-parameter view; default `schematic_nport`, recommended `EMX_spar`

`EMX_schematic_dc_name` the DC connectivity view; default `EMX_dc`.

Naming the views as recommended here lets you easily create multiple extracted views of each cell. You can then use the Virtuoso Hierarchy Editor to select different views for different cells according to the needs of your simulation.

Some advanced customization tricks are discussed in subsection 4.10.

The final line in `emxconfig.il` loads the rest of the interface files and should be left as-is. Once you have set up the configuration file, you just need to load it when Virtuoso starts. Add the following lines to a `.cdsinit` file in the directory where Virtuoso is started:

```
;;; EMX initialization start
(load "/path/to/emxskill/emxconfig.il")
;;; EMX initialization end
```

(Replace the path to `emxconfig.il` as appropriate.) When `emxconfig.il` is loaded, it will automatically load the rest of the interface.

If you use the model creation facility (see subsection 3.9), then you must also include the library `EMX_models` in your `cds.lib` or `lib.defs` (whichever is appropriate for your Cadence version). This library is located in the directory given by `EMX_interface_path`.

2 Invoking the Interface

When loaded, the interface installs a new menu bar item `EMX` for layout windows, as shown on the top right of figure 5. Click on `EMX` and select `Simulate` to bring up the main form, shown in figure 6. There is also a second form that is shown when you click the `Advanced options` button on the main form. The advanced form is shown in figure 7. The *EMX Virtuoso Interface Tutorial* discusses the usual sequence of steps for simulating a layout, but here is an outline.

1. Set the process information, if needed.
2. Select black-box cells, if using hierarchical simulation.
3. Check the layout that will be simulated by EMX.
4. Set the meshing options.
5. Fill in port names.
6. Set the desired frequency range.
7. Set whatever advanced options you desire.
8. Run EMX.
9. Plot the results.
10. Create a circuit model, and check how the model compares to the EMX simulation.

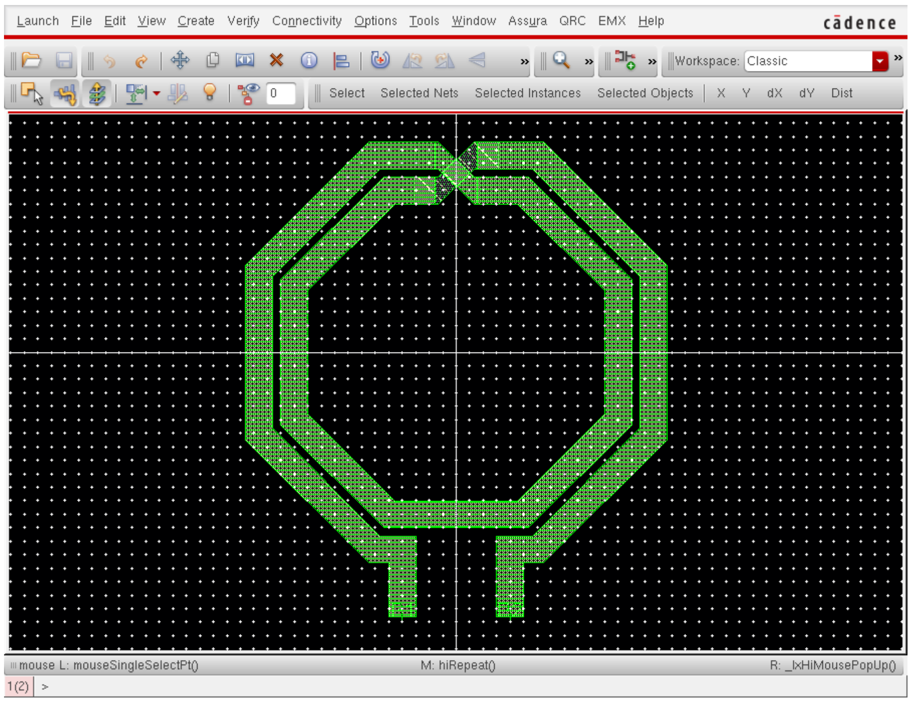
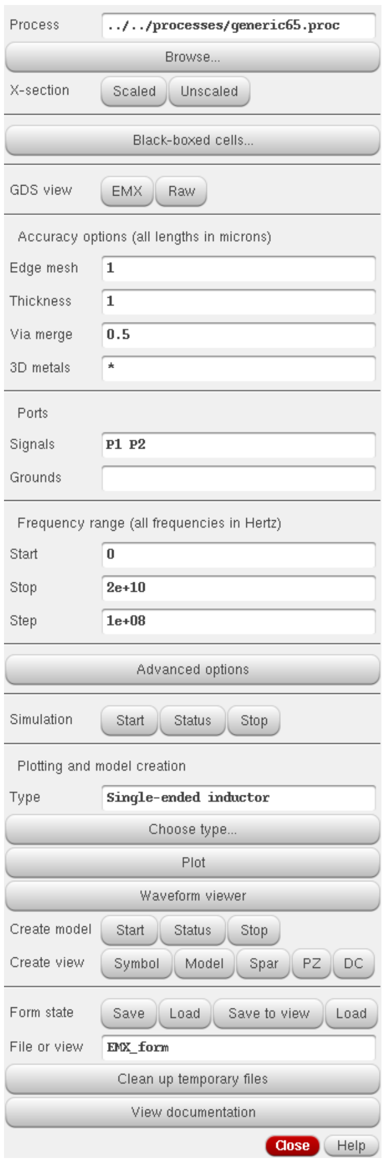
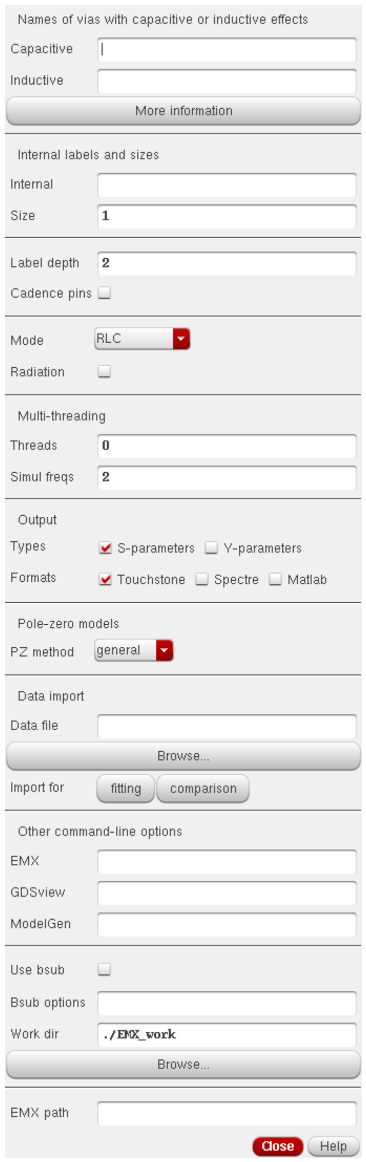

Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC EMX Help
cadence
Workspace: Classic
Select Selected Nets Selected Instances Selected Objects X Y dX dY Dist
mouse L: mouseSingleSelectPt()
M: hiRepeat()
R: _lxHiMousePopUp()
1(2) >

Figure 5: Inductor layout showing EMX menu item



Process	../../processes/generic65.proc
Edge mesh	1
Thickness	1
Via merge	0.5
3D metals	*
Signals	P1 P2
Grounds	
Start	0
Stop	2e+10
Step	1e+08
Type	Single-ended inductor
File or view	EMX_form

Figure 6: Main interface form



Names of vias with capacitive or inductive effects

Capacitive:

Inductive:

Internal labels and sizes

Internal:

Size: 1

Label depth: 2

Cadence pins: ☐

Mode: RLC

Radiation: ☐

Multi-threading

Threads: 0

Simul freqs: 2

Output

Types: ✓ S-parameters ☐ Y-parameters

Formats: ✓ Touchstone ☐ Spectre ☐ Matlab

Pole-zero models

PZ method: general

Data import

Data file:

Import for

Other command-line options

EMX:

GDSview:

ModelGen:

Use bsub: ☐

Bsub options:

Work dir: ./EMX_work

EMX path:

Figure 7: Advanced interface form

11. Create symbol and extracted views.

These steps correspond to the fields on the main form, reading from top to bottom. The rest of this manual discusses the fields on the main and advanced forms in detail.

3 The Main Form

3.1 Process information

The process information is shown in figure 8. The `Process` text box has the filename of the EMX process file. You can type it directly, or you can use Virtuoso's file browser to select the file by clicking the `Browse...` button. The default process file comes from the first file with the extension `.proc` in the process directory (given by `EMX_process_path` in the configuration file).

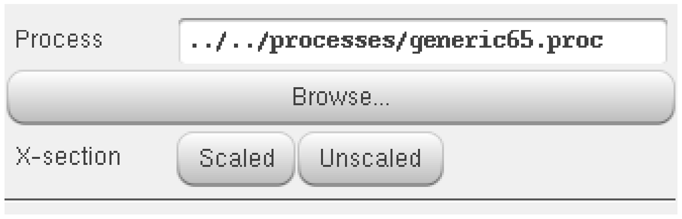


Figure 8: Process information

When you are editing the contents of the process file, it is sometimes useful to look at a cross-section view. Click the `Scaled` and `Unscaled` buttons to display a cross-section. The `Scaled` view artificially stretches layers and metals so that they can be labeled more clearly, while the `Unscaled` view shows layers and metals in proportion.

3.2 Choosing black-box cells

The `Black-boxed cells...` button is only displayed when the `EMX_black_box` option in the `emxconfig.il` file is set to `t`. Black boxes are only supported in EMX versions 4.0 and later. See subsection 3.13 for details on black box simulations.

Figure 9: Black-boxed cells button

3.3 Checking what EMX will simulate

The section shown in figure 10 has two buttons for looking at the GDSII layout. Both invoke the GDSview layout display program that is distributed with EMX. For details of how to control GDSview, see the *GDSview User's Manual*.

The first button, `EMX`, shows the layout that EMX will simulate. This view takes into account all effects specified in the EMX process file, such as via merging, slot elimination, metal biases, masking, or other layer operations. The layers shown correspond to the conductors defined in the process file.

The second button, `Raw`, displays the GDSII file as streamed out by Virtuoso. The layers displayed by GDSview will have the form `LxxTyy`, where `xx` and `yy` are the GDSII layer and type numbers.

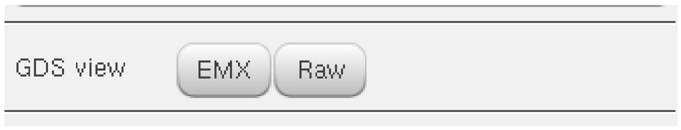


Figure 10: GDSview section

3.4 Meshing options

EMX's basic meshing controls are set by the fields shown in figure 11. These fields correspond directly to EMX options:

`Edge mesh` sets EMX's `--edge-width` (`-e`) option. It determines the width of mesh elements at the edge of conductors.

`Thickness` determines the EMX `--thickness` (`-t`) option. Conductors are split vertically if they are thicker than this size.

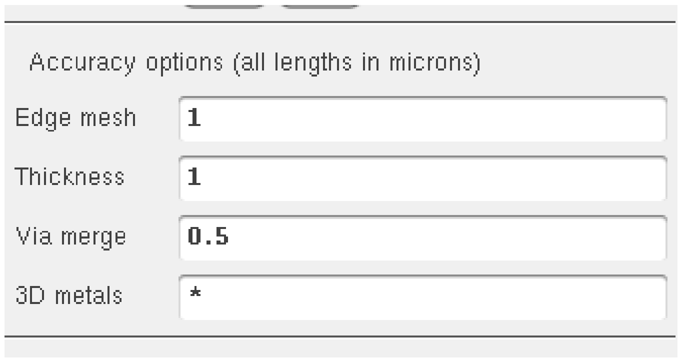
`Via merge` controls EMX's global via merging (`--via-separation` or `-v`). If you're controlling via merging on a layer-by-layer basis in the process file, you should set this field to `0`.

`3D metals` sets the `--3d` (`-3`) option. If you leave this field blank, then EMX uses 2D approximations for all conductors. If you set it to `*`, then all conductors will be treated as fully 3D. Or you can give a list of conductor names, like `me7 me8 me9`, if you want to use 3D for only certain metals.

For a more detailed explanation of how EMX works and what these options control, see the *EMX User's Manual*.

3.5 Ports

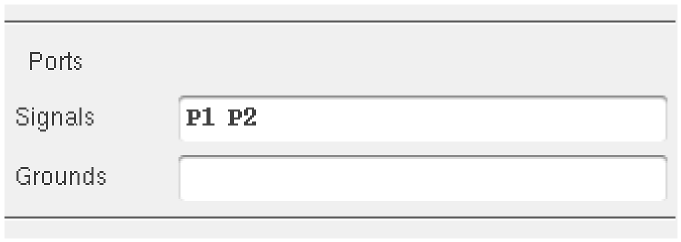
You tell EMX where you will make connections to the layout in the `Ports` section, shown in figure 12. There are two basic ways of specifying ports: by using plain labels, or by using Virtuoso's pins. The latter is recommended since


Accuracy options (all lengths in microns)

Edge mesh	1
Thickness	1
Via merge	0.5
3D metals	*

Figure 11: Meshing control section

it is a bit more flexible and integrates better with the Cadence environment, but we will discuss both methods.


Ports

Signals	P1 P2
Grounds	

Figure 12: Ports section

If you choose to use plain labels, then you should place them in the layout at places where you want connections. EMX supports both edge and internal connections. To make an edge connection, put the label in the middle of the edge where you want the port. To make an internal connection, put the label at the interior point where you want it to connect and add it to the `Internal labels and sizes` section of the advanced form (see subsection 4.2). In both cases, the label must be on a layer that is mapped to the appropriate conductor in EMX's process file. For example, if you want a connection to the conductor `metal5`, then you could use the `metal5 drawing` layer in Virtuoso for the label. Alternatively, if your process file defines EMX's `metal5` as the combination of Cadence's `metal5 drawing` and `pin` layers, then you could also put the label on `metal5 pin`.

Once you have placed the labels, then you should list the labels where you want connections in the `Signals` and `Grounds` fields. Each of these fields is a list of label names. `Signals` gives the ports that EMX will actually drive, while the labels given in `Grounds` will have connections but will be held at zero voltage. The number of ports in EMX's output is equal to the number of labels in the `Signals` field. With the settings shown in the figure, EMX will drive the connections at labels `P1` and `P2` and will produce a two-port output. There are no connections that are held at ground in this example.

The order of ports in the EMX results is the same as the order listed in the `Signals` line. If you want to use the plotting and model creation facilities for specific types of devices, then the order must be the same as the order assumed by that device. See subsection 3.10 for a list of the supported devices and their port orders.

If you want to use Virtuoso's pins to make connections, then you should follow these conventions:

1. Use rectangle pins to indicate where you want connections.
2. Specify the pin name in the `Signals` or `Grounds` section.
3. The access direction of the pin determines whether you want an edge or internal connection.
4. For an edge connection, align one edge of the pin rectangle with the conductor edge where you want the connection. Set the access direction of the pin to indicate that edge.
5. For an internal connection, set the access direction of the pin rectangle to `any` (i.e., `top`, `bottom`, `left`, and `right`). The connection is made within the entire area of the pin rectangle.

If you make the pin on a layer such as `me7 pin` rather than `me7 drawing`, be sure that the EMX process file uses both the drawing and pin layers when defining the conductor for `me7`. When using pins to make connections, you must also check the `Cadence pins` option on the advanced options form. We recommend putting all desired pins at the top level and setting the label depth on the advanced form to 0.

If you are using pins and making only n-port models, then there is no required port order. Then it is sometimes convenient not to have to list all of the port names explicitly, especially if the number of ports is very large. In this case, you can use a `*` in either the `Signals` or the `Grounds` field to mean "all other pins that are not explicitly mentioned." For example, if you set the `Signals` field to `*` and leave `Grounds` empty, then you'll get an n-port with a driven connection for each pin. If you have `*` for `Signals` and add the pins `GND1` and `GND2` to `Grounds`, then all the pins except `GND1` and `GND2` will be driven. Or if you put `P1` and `P2` for `Signals` and `*` for `Grounds`, then two pins will be driven and all others will be grounded. The unmentioned pins which `*` expands to are sorted alphabetically. So if you have a group of pins like `P1`, `P2`, ... and you want them in numeric order, be sure to include leading zeros if there are more than ten of them: `P01`, `P02`, ...

There is some additional flexibility in the `Signals` field:

1. You can write complex port specifiers, such as

   ```
   S1:G1,G2,G3 S2:G1,G2,G3
   ```

This might correspond to a GSGSG probe structure, with the first port connected between signal pad `S1` and the grounds `G1`, `G2`, and `G3`, and the second port using signal pad `S2` and the same grounds. See the *EMX User's Manual* for more information on port specifiers. Generated symbol and extracted views in this case will have the complex port specifiers for pin names.

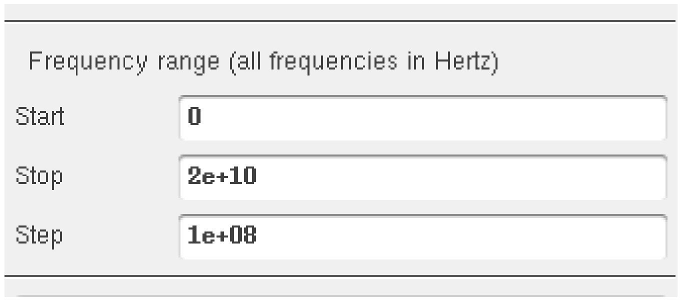
2. You can indicate that a pair of ports are typically driven differentially by separating them with a semicolon. For example, `PLUS;MINUS` would indicate that the port voltages are usually in anti-phase. The number of ports and the symbol and extracted views are the same as if you had just written `PLUS MINUS`, but EMX is run with an extra `--mode=PLUS-MINUS` option. This can improve simulation accuracy in some cases; see the *EMX User's Manual* for more details.

If there are no ground signals, then the reference pin in the symbol and extracted views is called `GND`. When you specify exactly one ground signal, then that signal's name is used for the reference pin. If there are multiple grounded signals and one is called `GND` or `gnd`, then that one is used for the reference. Otherwise, the first grounded signal is used as the name of the reference.

The `Signals` and `Grounds` fields also support bus syntax. For example, `in<0:2>` expands to `in<0> in<1> in<2>`. If you do this for the `Signals` field, the generated views (symbol, S-parameter, pole-zero model, etc.) will have a bus pin as the external connection.

3.6 Frequency range

Set the frequency range for the simulation with the fields shown in figure 13. If you want to simulate just a single frequency, set the `Start` and `Stop` values to the same frequency. In this case, the `Step` field is ignored.


Frequency range (all frequencies in Hertz)

Start	0
Stop	2e+10
Step	1e+08

Figure 13: Frequency section

3.7 Advanced options

The `Advanced options` button displays the form in figure 7. The fields on that form are described in section 4.

Figure 14: Advanced options button

3.8 Simulation control

Use the buttons shown in figure 15 to control the simulation. Click `Start` to being a simulation. This will also open a window that displays the simulation progress. If you close that window before the simulation finishes, then you can get another log window by clicking `Status`. And if you want to interrupt the simulation for some reason, you can do so by clicking the `Stop` button. You can only run one simulation on the same cell at a time, but it is possible to run different simulations on different cells simultaneously.

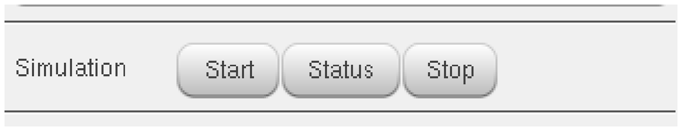


Figure 15: Simulation control buttons

3.9 Plotting and model creation

Figure 16 shows the controls for plotting, making models, and creating symbol and extracted views. Plotting and model creation is only supported for certain types of devices (but see subsections 3.11 and 3.12). If the layout that you are working on matches one of these types, then you should pick it from a menu by clicking the `Choose type...` button. Then `Plot` will display a window with appropriate electrical parameters. There is no specialized plotting for the `N-port` device type.

You can also click the `Waveform viewer` button to display Cadence's simulation results browser and do your own plotting. In this case, you can find the simulation results in the directory set by `EMX_working_dir` from the `emxconfig.il` file. They will be in a subdirectory called `library_cell.work/cell.raw`, where the `library` and `cell` are replaced by the library and cell names.

For all of the device types except for `N-port`, you can request a lumped circuit model by clicking the `Start` button next to `Create model`. This starts

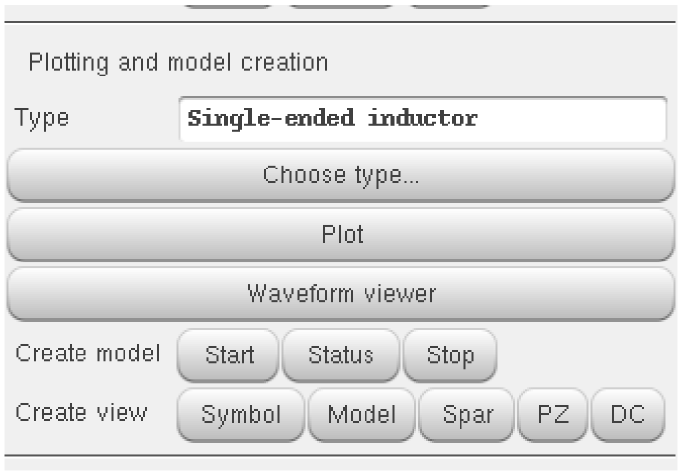


Figure 16: Plotting and model creation controls

a model creation job using the ModelGen program and gives you a log window, just like when you start a simulation. You can get back to a closed log window by clicking `Status`, and clicking `Stop` will interrupt the model creation.

The generated model is not guaranteed to exactly match the electrical behavior of the actual layout. To see how good the model is, click `Plot` again after creating the model. The plot will show the simulation results compared to a playback of the model.

After making a model, use the buttons under `Create view` to produce different views. The `Symbol` button creates a symbol matching the selected device type. (It also creates matching `auCdl` and `auLvs` views if the appropriate options are set in `emxconfig.il`.)

The `Model` button makes an extracted schematic view for the lumped model. If you look at that view, you will see an instantiation of a cell in the `EMX_models` library. The parameters of that instantiation (which you can view by examining the instantiations properties) correspond to the component values for the circuit topology corresponding to the device. When simulating circuits containing the cell, you need to include a model library that contains the different topologies. For Spectre, that model library is located in the interface installation directory, in the file

```
EMX_spectre/emxmodels.scs
```

For Hspice, include `EMX_hspice/emxmodels.sp` instead.

Use the `S-params` button to make an extracted schematic view that directly references the frequency-domain simulated S-parameters. By default, the reference to the S-parameters is via an absolute file path, but see the `EMX_nport_sparam_view_name` and `EMX_nport_sparam_view_prefix` configuration variables for other possibilities. If your simulator handles frequency-domain models well, then using the `S-params` view will give you the best accuracy.

The `PZ` button creates an extracted schematic view for a pole-zero model (see subsection 3.11).

`DC` creates a schematic view that contains resistors which reproduce the DC connectivity information (this is the same information that is shown by the `--print-connectivity` option in EMX). This is mostly useful for LVS when using the black box flow (see subsection 3.13).

When using the black box flow it is not possible to make lumped models. Only the S-parameter, pole-zero, and possibly DC views are available, and the extracted views will contain instances of the black-boxed cells.

3.10 Device types

Here is a list of all of the device types and the assumed port ordering.

`Single-ended inductor` is a two-terminal inductor. The port order is `PLUS`, `MINUS`. For plotting, the `MINUS` terminal is assumed to be grounded.

`Differential inductor` is the same except that `PLUS` and `MINUS` are assumed to be operating in differential mode for plotting.

`Single-ended shield inductor` is like the normal singled-ended inductor but there is a shield under the inductor to isolate it from the substrate. The port order is `PLUS`, `MINUS`, `SHIELD`. For plotting, the shield is assumed to be grounded.

`Differential shield inductor` is the same but with `PLUS` and `MINUS` driven differentially for plotting.

`Tapped inductor (diff mode only)` has terminals `PLUS`, `MINUS`, `TAP`. In normal operation, the `TAP` has a DC voltage and `PLUS` and `MINUS` are driven differentially. The model fitting emphasizes accuracy only for differential mode.

`Tapped inductor (common mode too)` is similar, but the model creation gives more emphasis to also trying to match common-mode behavior.

`Tapped shield inductor (diff only)` has both a shield and a center tap. The port order is `PLUS`, `MINUS`, `TAP`, `SHIELD`.

`Tapped shield inductor (comm too)` is the same, but model creation emphasizes common-mode behavior more.

`Single-ended cap (symm)` is a two-terminal capacitor. The terminals, `PLUS` and `MINUS`, are approximately symmetric, as is typical with inter-digitated capacitors. Plotting assumes that `MINUS` is grounded. The parasitic substrate capacitance is symmetric.

`Differential cap (symm)` is the same, but the plotting is for when the terminals are operated differentially.

`Single-ended cap (asymm)` and `Differential cap (asymm)` are similar but with an asymmetric substrate.

`MiM capacitor` is a two-terminal capacitor that is asymmetric. The port order is `TOP`, `BOTTOM`. The topology assumes that `TOP` has negligible substrate capacitance.

`Xformer, no taps (no comm)` is a four-terminal transformer. The port order is `P1`, `P2`, `S1`, `S2`. The "dot" convention is for the dots to be on `P1` and `S1`. For plotting, `P2` and `S2` are assumed to be grounded. The model fitting does not emphasize accuracy of common-mode behavior.

`Xformer, primary tap (no comm)` is a five-terminal transformer with the primary terminal tapped. The ports are `P1`, `P2`, `S1`, `S2`, `PTAP`. For plotting, `PTAP` and `S2` are grounded, and `P1` and `P2` are operated differentially.

`Xformer, secondary tap (no comm)` is similar but the center tap is on the secondary side. The ports are `P1`, `P2`, `S1`, `S2`, `STAP`.

`Xformer, dual-tapped (no comm)` has ports `P1`, `P2`, `S1`, `S2`, `PTAP`, `STAP`.

`Xformer, no taps (comm too)`
`Xformer, primary tap (comm too)`
`Xformer, secondary tap (comm too)`
`Xformer, dual-tapped (comm too)`
These are like the previous models, and have the same port order. Some weight is given to common-mode operation in fitting.

`Symm xformer, no taps (no cm)`
`Symm xformer, primary tap (no cm)`
`Symm xformer, secondary tap (no cm)`
`Symm xformer, dual taps (no cm)`
`Symm xformer, no taps (comm)`
`Symm xformer, primary tap (comm)`
`Symm xformer, secondary tap (comm)`
`Symm xformer, dual taps (comm)`
These are versions of the above transformer models, but the model is forced to be completely symmetric.

`Shield xformer, no taps (no cm)`
`Shield xformer, primary tap (no cm)`
`Shield xformer, 2ndary tap (no cm)`
`Shield xformer, dual taps (no cm)`
`Shield xformer, no taps (comm)`
`Shield xformer, primary tap (comm)`
`Shield xformer, 2ndary tap (comm)`
`Shield xformer, dual taps (comm)`
These are like the previous transformer models, but they are for devices

with shields. The port order is the same as with the other models, but there is an additional `SHIELD` port at the end.

`Tcoil (simple model)` is a three-terminal device, basically an inductor with a tap somewhere along it, though there is no assumption about differential operation. The port order is `P1`, `P2`, `TAP`.

`Tcoil (complex model)` is similar, but uses a more complicated model topology.

`Shield tcoil` is like these models, but also has a `SHIELD` port at the end of the order.

`Transmission line` has two-terminals, `P1` and `P2`. The line is assumed to have relatively low loss, and the model is a series of RLC sections.

`Diff transmission line` is like the transmission line model but with two coupled lines. One has ends `P1` and `P2`, and the other has ends `P3` and `P4`.

`Two wires` (simple `interconnect` model) has four terminals, with a simple RL series combination between `P1` and `P2`, and another RL combination between `P3` and `P4`. There are also all possible capacitive couplings between the ports without DC connections, and each port has a coupling to a simple substrate model. `Three wires` and `Four wires` are analogous, but with more pairs of terminals. These models are intended for small groups of wires that are mostly dominated by RC effects, such as the connections in a transistor.

`N-port` can have any number of ports. There is no special plotting or model creation, so there are no requirements on the port order.

3.11 Pole-zero models

More recent versions of EMX can create pole-zero (a.k.a. state-space) models directly. This ability is still experimental. It has some advantages and some disadvantages compared to the usual model creation as described in subsection 3.9.

1. The pole-zero models do not assume any particular topology, so models can be made for any type of device.

2. The pole-zero models for complicated devices such as tapped transformers may be more accurate than the fixed-topology models, though this is not guaranteed. While fixed-topology models are constructed using objective functions appropriate to the type of device, the pole-zero models only try to fit the device S-parameters. For high-Q structures, the default S-parameter fitting tolerance may not suffice to make a good model. There are options in EMX to adjust the tolerance, but it is not guaranteed that EMX will be able to make an arbitrarily accurate model.

3. The pole-zero models are generally larger than the fixed-topology models. They contain controlled sources, not just RLCK elements. They should be passive and have correct noise behavior however.

4. The complexity of constructing pole-zero models goes up rapidly with the number of poles required and the number of ports of the device. The time required for making a pole-zero model can be excessive in some cases.

In order to create a pole-zero model, you first need to make sure that the `EMX_pole_zero_models` in `emxconfig.il` is set to `t` (or use a recent version of EMX that supports auto-configuration). Make sure that the `PZ method` field on the advanced options form is set to `general` or `DC only`. Then run your EMX simulation. You must do a sweep to make a pole-zero model. In most cases, we recommend that the sweep start at zero, otherwise the model will be an open circuit at DC. (Note that starting at zero is *not* the same as using `--include-dc`; that option does a separate discrete DC point which is not part of the sweep.) After EMX finishes computing the frequency sweep, it will try to create a model, using increasing number of poles until the required accuracy is achieved. At that point you can use the interface's plotting facilities just as with the fixed-topology models to judge the quality of the pole-zero model.

The difference between the `general` and `DC only` methods is that the latter assumes that the process is frequency-independent (or that you've forced it to be frequency-independent by choosing the `RLC @ DC` or `RC @ DC` simulation type). When the process is frequency-independent, a more efficient modeling method (corresponding to EMX's `--model-reduce-only` option) can be used. When the process is not frequency-independent, then choosing `DC only` will generally just make the pole-zero modeling fail to converge.

If the model is not accurate enough, you can adjust the tolerances by giving options for EMX in the `Other command-line options` section of the advanced form. To try different tolerances without re-running the sweep from the beginning, add `--load-model-state` to the options, e.g., like this:

```
--load-model-state --model-tol=3e-5
```

Then when you click the button to start the simulation, only the pole-zero modeling step will be repeated. For a list of all the options controlling model creation, see the *EMX User's Manual*.

If you want to run a sweep without creating a pole-zero model but also want to have the ability to try a pole-zero model later, add `--save-model-state` to the command-line options. This option is automatically included when `PZ method` is not `none`. The model state is saved before model creation begins, so if you interrupt the simulation during model creation you can still use `--load-model-state` later.

After you are satisfied with the model, click `PZ` to make a pole-zero extracted view. If you are using only pole-zero models, then you do not need to include the EMX model library (`emxmodels.scs` or `emxmodels.sp`) when running simulations.

3.12 User-defined model topologies

New versions of ModelGen support user-defined model topologies. To use this through the interface, you need to create a template schematic whose topology matches that of the desired model. For example, a simple topology for an inductor model is shown in figure 17. In the template schematic, observe the following points:

1. There should be a pin for every port in the model. The ports can be called whatever you like, except the reference port must be called `GND`. You must set the pin order property for the schematic to match the signal order in the EMX simulation. `GND` must be the last pin in the order.

2. The allowed components are in `analogLib`: `res`, `cap`, `ind`, and `mind`. Most components will have a simple value that indicates the approximate expected order-of-magnitude of the value. For mutual inductors, set the `k` value to the maximum magnitude desired (so for an unrestricted coupling element, use 1).

3. If you want a component to have a value that is equal to the value of a second component then use the name of the second component as the value. For example, `C2` in the figure should have the same value as `C1`.

4. If you want a component to have a particular value, set the value field to `fixed ...`, where the `...` is a floating point number (no suffixes or units are allowed). In the example, `C0` is a 10 fF capacitor.

5. To limit the range of a component, set the value to something of the form `[..., ...]`, where the `...` are lower and upper bounds (again, floating point numbers with no suffixes or units).

6. There must be a text label of the form `objective: ...` where `...` is the name of a ModelGen type. You can find all the available types with the command `modelgen --help`. Alternatively, you can select a device from the menu and then copy the value of the `Type` field and use that for `...`

Once you've created the template schematic, then you should fill in the `Type` field with with a string of the form `user ...`, where the `...` is replaced by the library name, cell name, and view name of the template. For example, if your template is the `schematic` view of cell `indtopology` in library `mylib`, then use

```
user mylib indtopology schematic
```

as the value of the `Type` field.

Plotting works for user-defined models if (and only if) the type (from the `objective`) field corresponds to one of the built-in devices.

After you run ModelGen, clicking the `Model` view will make a copy of the template schematic, rename the ports appropriately, and fill in the specific values of the components. For example, producing a model from an inductor using the topology in figure 17 might produce the schematic in figure 18.

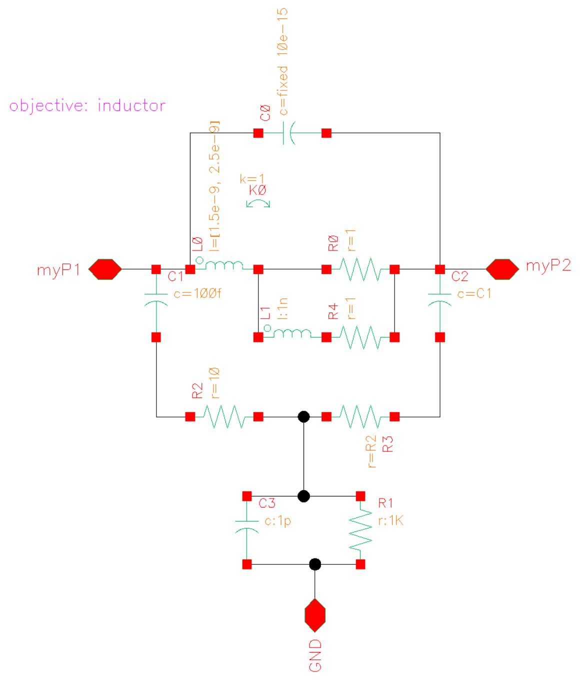


Figure 17: Simple template schematic for an inductor

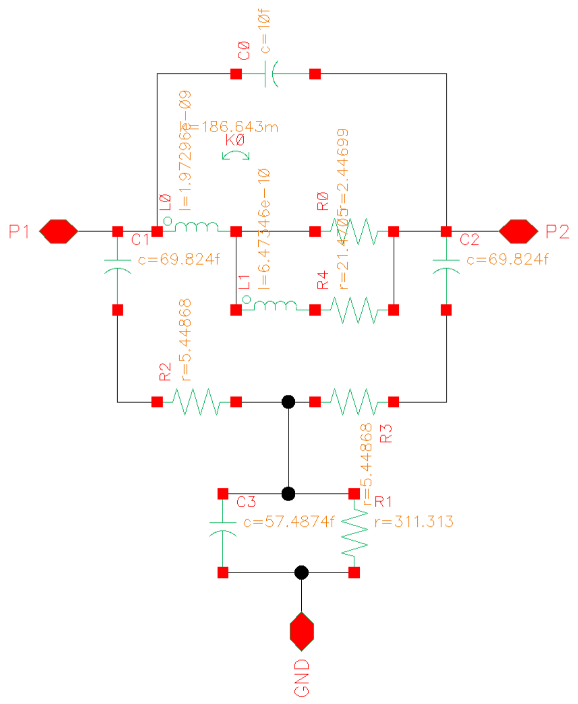
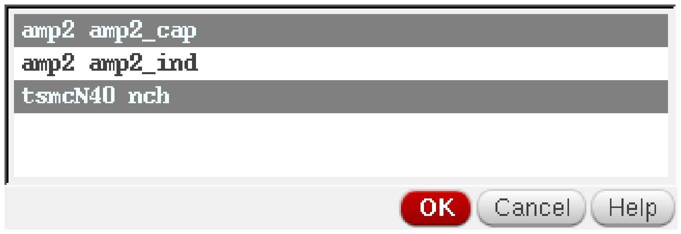


Figure 18: Generated schematic for a model of a specific inductor

3.13 Black-box simulations

When you click the `Black-boxed cells...` button, then a selection list will appear (figure 19). The listed cells are all those subcells within your layout that have symbol views. You can select a cell by clicking, select a range of cells by shift-clicking, and toggle a cell's selection by control-clicking.



amp2 amp2_cap
amp2 amp2_ind
tsmcN40 nch

Figure 19: Black-boxed cells selection list

Any cells that you select will be treated as black boxes during the simulation. That means that EMX will not look at the contents of the cells except for any top-level pins. (If you're not using pins, then top-level labels are used for box terminals instead.) Those pins will become extra ports in the simulation. When you create an extracted schematic view (either an S-parameter view, a pole-zero model view, or a DC connectivity view) then the view will also include instances of the omitted black-boxed cells.

As an example, consider the amplifier whose schematic is shown in figure 20 and whose corresponding layout is shown figure 21. Suppose that you want to simulate the routing and inductor of the amplifier, while black boxing the transistors and the capacitor. You would select the black-boxed cells as shown in figure 19.

The GDSview display of what EMX will simulate is shown in figure 22. Note that the positions of the transistors and the capacitor are indicated by their bounding boxes (labeled by `nch` and `amp2_cap`). EMX will produce S-parameters for the rest of the layout, including the inductor.

After the simulation is finished, generating an S-parameter view (subsection 3.9) gives the schematic shown in figure 23. The `n12port` block represents the S-parameters for the interconnect and the inductor, while the instances `C0`, `M0`, and `M1` are from the black-boxed cells.

The S-parameter and pole-zero views are the ones that you would use at the circuit simulation level. You can select which view to use with the Cadence configuration editor.

For this example, the design proceeded as follows:

1. Design the basic amplifier using ideal inductors and capacitors.

2. Choose real inductor and capacitor layouts using a scalable component library of pcells. Re-simulate with scalable electrical models replacing the ideal inductor and capacitor.

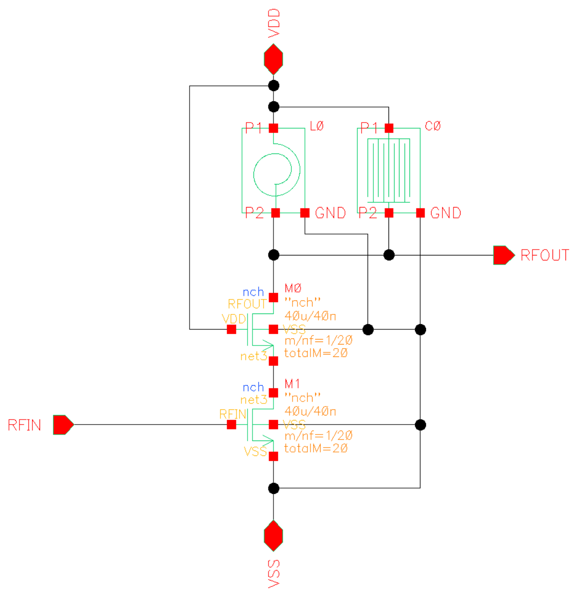


Figure 20: Amplifier schematic

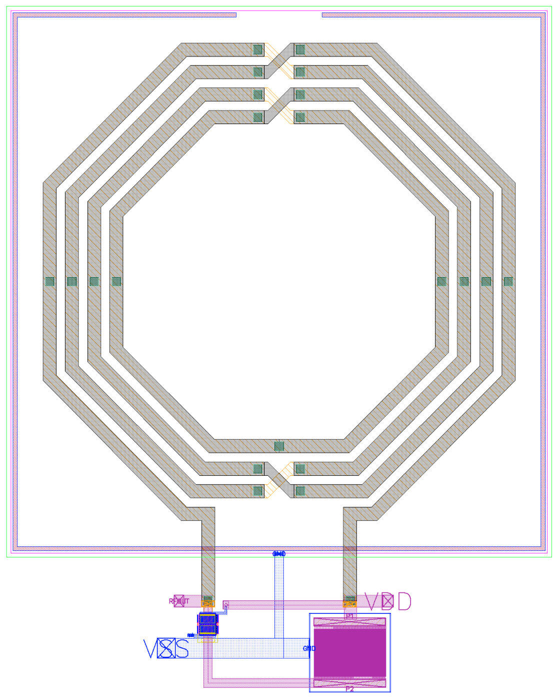


Figure 21: Amplifier layout

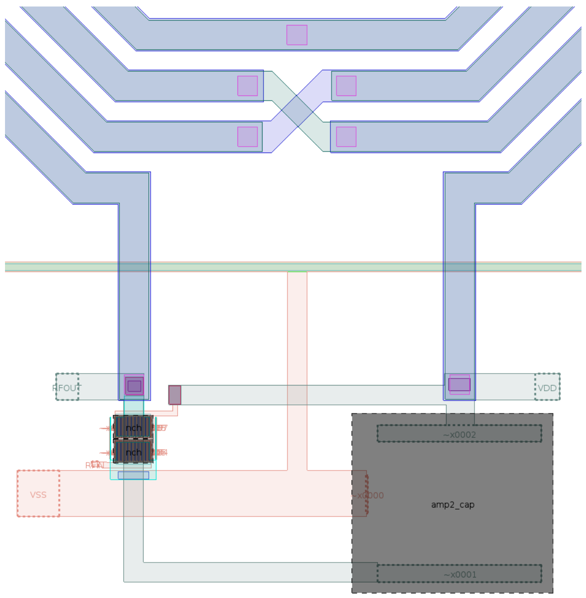


Figure 22: GDSview of the amplifier layout with black boxes

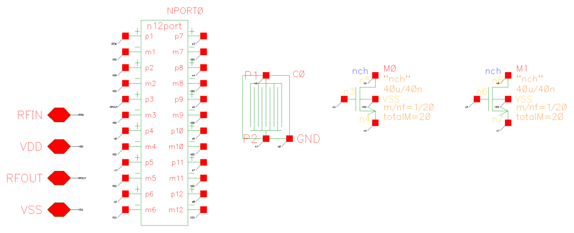


Figure 23: Amplifier S-parameter view with instances of black-boxed cells

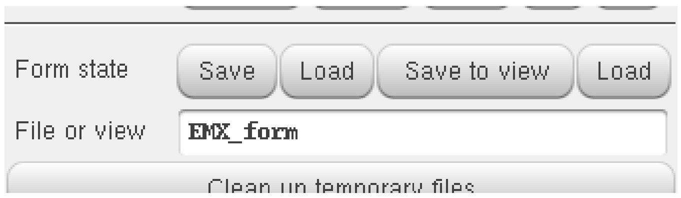
3. Run EMX on the specific inductor and capacitor and re-simulate.
4. Do the layout. Select all the cells as black-boxed cells. Run EMX at DC and generate a DC view, then run LVS (see below).
5. Select the capacitor and transistors as black-boxed cells. Run EMX and generate an S-parameter view. Re-simulate the final design.

The DC connectivity view is mostly useful for LVS. If you black-box all the primitive cells, and then run a simulation and generate the DC view, that view will reflect EMX's view of the primitive device connectivity. If this does not match the layout (or the original schematic), then it means that you haven't connected the primitives in a manner that is consistent with EMX's black-box flow. Because EMX does not look inside black-boxed cells except to get the locations of the cell terminals, you cannot rely on any other geometry within the cells for connectivity.

3.14 Saving form values

You can save the state of the main and advanced interface forms using the controls in figure 24. Clicking `Save` writes a text file with all the form values. The name of the file is set by the `File or view` field, and a `.txt` extension is automatically added if the name has no extension. The location of the file is always in the `library_cell.work` subdirectory of the `EMX_working_dir` directory. Click `Load` to restore the previously-saved form contents from this file.

Alternatively, you can save the state as a view associated with the layout. In that case, the view name is given by the `File or view` field, and the state is kept in file `text.txt` in the subdirectory for that view. The `Save to view` button and the final `Load` button save and load the state from the view.



File or view: EMX_form

Figure 24: Form state controls

3.15 Clean-up and viewing documentation

The final two buttons (figure 25) provide a means for cleaning up intermediate EMX files and for viewing documentation for the interface. (The second button only appears if a command for viewing the documentation has been defined in the interface configuration file.)

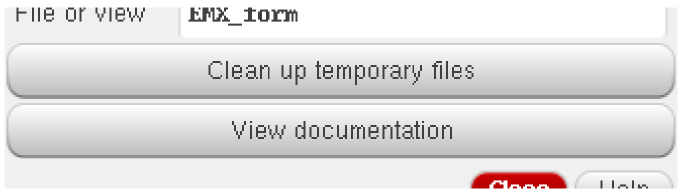


Figure 25: Clean-up and documentation buttons

4 The Advanced Form

The fields on the advanced form control less common options.

4.1 Via modeling

Normally, vias in EMX are modeled just as resistors. If you want to include sidewall capacitance or via inductance, then list the names of the appropriate vias in the `Capacitive` and `Inductive` fields shown in figure 26. For example, `via3 via4` will turn on capacitive or inductive effects for those two vias. Write a `*` to mean "all vias." Write `-via1 via2` to mean "all vias except `via1` and `via2`." Click `More information` for a summary of what's allowed.

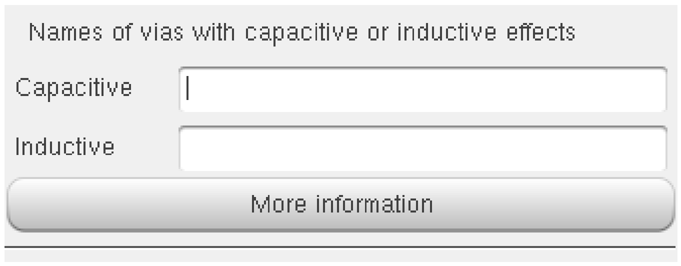
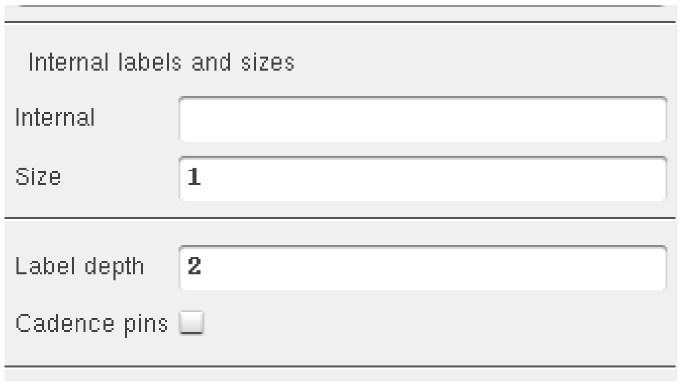


Figure 26: Via controls

4.2 Labels

Figure 27 shows the fields that control labeling. If you are using ordinary labels rather than Virtuoso's pins, then you must put the names of any internal labels in the `Internal` field. The internal labels will all be squares with size given in the `Size` field (by default, one micron on a side). EMX will see labels in subcells down to a depth in the hierarchy specified by `Label depth`. Depth 0 means only top-level labels, depth 1 means also consider labels in first-level subcells, etc.

If you are using pins, click the `Cadence pins` option. In this case, you should usually set `Label depth` to 0 so that EMX only sees pins belonging to the top-level cell. However, you can set the depth to higher values if you do want to



Internal labels and sizes

Internal:

Size: 1

Label depth: 2

Cadence pins: ☐

Figure 27: Label control fields

see pins in subcells for some reason. If you fill in the `Internal` and `Size` fields when using Cadence pins then the pin geometry may be overridden; this is legal but potentially confusing and is not recommended.

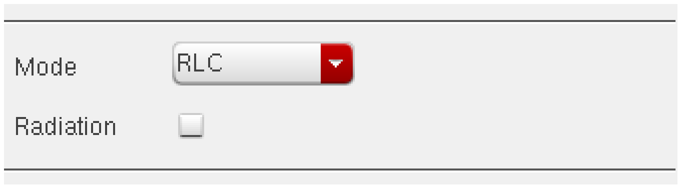
4.3 Simulation type

The fields of figure 28 control the type of simulation. Normally EMX models all resistive, inductive, and capacitive effects for conductors (`RLC` mode). Selecting `C only` will give you just a capacitance solve (though frequency-dependencies in the dielectrics are still accounted for). Selecting `RL only` makes EMX simulate only the resistance and inductance, and choosing `RC only` makes it simulate on resistance and capacitance. The `RLC @ DC` and `RC @ DC` options are only available with later versions of EMX. They make EMX use just the DC Green's functions, which effectively turns it into something like a static extractor. This reduces simulation time and memory (and also pole-zero modeling time if you set `PZ method` to `DC only`). The main difference in the case of IC processes is that the grounding effect of the substrate is stronger (e.g., inductors will generally resonate faster because of a larger parasitic substrate capacitance). The DC options are mostly intended for modeling the smaller parasitic effects of general interconnect, especially when using the black-box facilities (subsection 3.13).

By default, EMX does not model radiation effects, but you can make it do so by clicking `Radiation`. Correctly modeling high-frequency substrate losses with highly-conductive substrates (around 1000 S/m) also requires that you use `Radiation`.

4.4 Parallelism

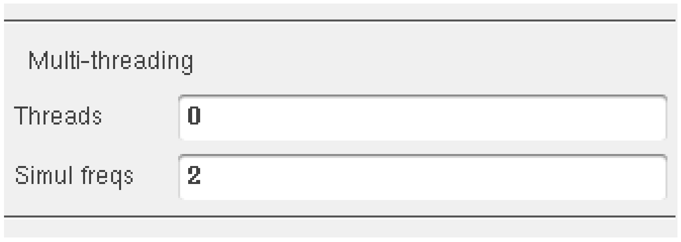
If you have a multi-threaded version of EMX, then you can control the parallelism with the fields of figure 29. The number of `Threads` should be about the same or a bit larger than the number of CPUs you want to use. If you


Mode: RLC

Radiation: ☐

Figure 28: Simulation type

set **Threads** to 0, then EMX will automatically choose the number of threads based on the number of CPUs in the machine. The number **Simul freqs** is the number of frequency points to solve for at once. Increasing this beyond 1 generally makes more efficient use of the available CPUs, but at the cost of increased memory. If you have sufficient memory, then try 2 or perhaps 3.


Multi-threading

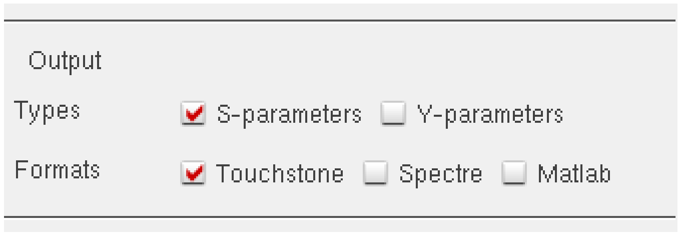
Threads: 0

Simul freqs: 2

Figure 29: Parallelism controls

4.5 Output files for frequency-domain data

Choose the types and formats of output files with the fields in figure 30. The possibilities are described in detail in the *EMX User's Manual*. The output files all go in the working directory for the cell (`library_cell.work` in the `EMX_working_dir` directory). You get all combinations of output type and output format. The extension for an output file depends on the format of that file. If you are using the `N-port` model type, be sure to leave `S-parameters` and `Touchstone` checked.


Output

Types: ✓ S-parameters ☐ Y-parameters

Formats: ✓ Touchstone ☐ Spectre ☐ Matlab

Figure 30: Output types

4.6 Pole-zero models

This check box appears only if you have set the `EMX_pole_zero_models` to `t` in `emxconfig.il`. When the method is set to `general` or `DC only`, models are created by EMX directly (a pole-zero model). For details about pole-zero models, see subsection 3.11.

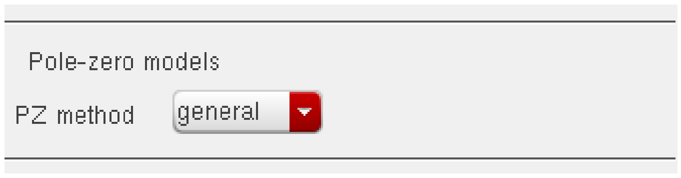


Figure 31: Pole-zero modeling

4.7 Data import

If you have data that you would like to compare with the simulation, you can import it for comparison (figure 32). The data should be in TOUCHSTONE format* Click `Browse...` to open a file browser for the file name. Once `Data file` is filled in, click `comparison` to import the file. When you `Plot` (subsection 3.9), the graphs will show the EMX simulation data as well as your imported data (plus the model if you created one). If you import the data for `fitting`, then model creation will use that data as well. You can use this to make circuit models of measurement data.

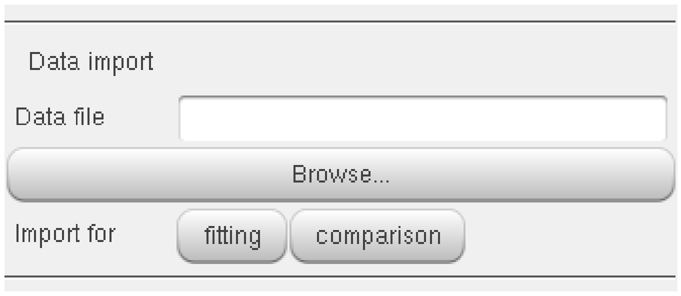
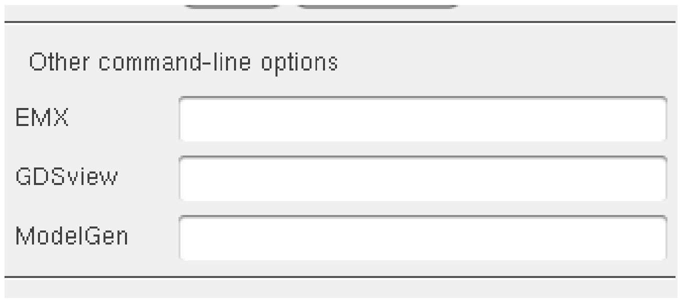


Figure 32: Data import fields

4.8 Other options

Give any additional command line options with the fields in figure 33. See the corresponding user's manual for possibilities.

*TOUCHSTONE is a registered trademark of Agilent Corporation.


Other command-line options

EMX

GDSview

ModelGen

Figure 33: Additional option fields

4.9 Job submission and paths

When the `EMX_remote_machine` variable is set to something that begins with `bsub` or `qsub` (or a path whose last component ending with one of those commands, e.g., `/path/to/bsub`), then the interface will try to submit jobs using the batch submission facility. The remainder of the `EMX_remote_machine` string will be used as an initial value for the `Bsub options` field (typically this part of the string would be used for things like the queue name). Check the `Use bsub` box to enable batch job submission. Batch submission may be further configured by setting these variables in `emxconfig.il`:

`EMX_bsub_default` Set this to `t` have the `Use bsub` field default to checked.

`EMX_viewproc_local` Normally the process cross-section viewing runs EMX locally to generate a PostScript file. Setting this variable to `t` makes EMX use the batch facility for `emx` in this instance also.

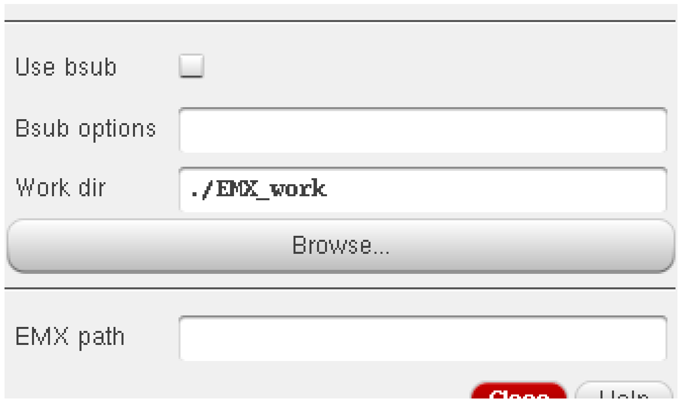
`EMX_spectre_local` Set this to `t` to make the interface run the Spectre circuit simulator (used for playing back simulation results) via the batch facility.

`EMX_modelgen_local` Set this to `t` to run `ModelGen` remotely via the batch facility.

At sites that have the Platform LSF facility for batch processing, you can use the check box and `Bsub options` field to submit EMX jobs via `bsub` rather than running them on your local machine. The options field would typically give the queue name. Note that the `EMX_remote_machine` variable in `emxconfig.il` must be set to a string that begins with `bsub`, otherwise these fields have no effect. Also, while the job's output is automatically captured and will be displayed in the simulation progress window, it usually happens with a delay. So initially the window will only show that the job was started; the actual output from EMX appears after some time.

The last fields of figure 34 can be used to override the working directory and the path to the EMX installation. Normally these are set in the `emxconfig.il` file, but you can override them here for things like testing a new version of EMX.

(But note that if you're using auto-configuration, that happens only when the interface is loaded, not when you change this field.)


Use bsub

Bsub options

Work dir ./EMX_work

EMX path

Figure 34: Paths

4.10 Advanced customization

In general you can use whatever Skill code you like within `emxconfig.il`. This means that you can set the interface configuration variables according to the shell environment (the `getShellEnvVar` function to get the value of a variable), according to shell commands (run a process with `ipcBeginProcess`, read output with `ipcReadProcess`), or according to other features of your Cadence installation or the design.

If you want to adjust the form values (or even form fields) whenever a new interface form is created, there is also variable `EMX_form_init_proc` that you can set to a symbol that names a general Skill function. If this variable is not `nil` and it is bound to a procedure, then the corresponding function is called when you first open the EMX simulation form for a cell. The procedure gets two arguments: the first is the main form (figure 6) and the second is the advanced form (figure 7). As an example, the following code in `emxconfig.il` would set the initial value for the model type to `N-port` and enable pole-zero model generation by default:

```
...
procedure(set_emx_form_vals(mainform advform)
  mainform->Type->value="N-port"
  advform->PZ_method->value="general"
)

EMX_form_init_proc='set_emx_form_vals
...
```