PassiveLib

User Manual

info@passivelib.com

November 6, 2021

Overview

Technology file

Technology file

- Technology file describes mapping GDSII layers to physical metal and via layers
- Comments are followed after #:

Comments

this is a comment

- Technology grid is set with the **grid** command:

Technology grid

grid = 0.005

- Metal and via layers are described within the structure **layer ... endLayer**

Technology file

- Metal layer:

Metal layer

layer M1 **metal**	# metal name is M1
gdsNum = 5	# gds number is 5
gdsType = 0	# gds type is 0
minS = 0.5	# minimal space between M1 layers
minW = 1.3	# minimal width
dSub = 206.5	# distance from substrate bottom plate
metT = 0.5	# metal thickness
endLayer	# end of layer command

- Metal name can be arbitrarily chosen
- **dSub** is distance between substrate bottom plate and layer bottom plate

Technology file

- Via layer:

Via layer

```
layer V12 via            # via name is V12
    gdsNum = 15          # gds number is 15
    gdsType = 1          # gds type is 1
    topMet = M2          # via is placed between M1 and M2
    botMet = M1          # via is placed between M1 and M2
    viaEnc = 0.05        # via enclosure with topMet/botMet
    viaSize = 0.2        # rectangle viaSize x viaSize
    viaSpace = 0.2       # minimal space between vias
endLayer                 # end of layer command
```

- Via name can be arbitrarily chosen
- Only square vias **viaSize** x **viaSize** are supported

Technology file

- Technology file should be imported with command line option or with environment variable

command line

--tech-file-name=tech.txt

environment variable

export PASSIVE_LIB_TECHNOLOGY="tech.txt"

General commands

General commands

- Print help, short and long options

print help

-h

--help

- Long option can be used with argument to print commands for the given component

--help=inductor-symmetric

- Create file with system information required for licensing PassiveLib

--host-id

General commands

- License file is activated with the command line option or with the environment variable

command line

```
--license-file-name=lic.txt
-l lic.txt
```

environment variable

```
export PASSIVE_LIB_LICENSE="lic.txt"
```

- Path to installed PassiveLib

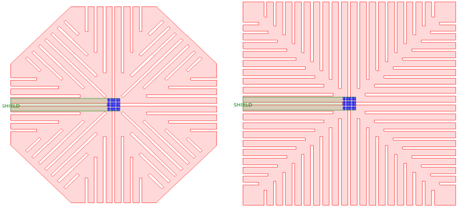
environment variable

```
export PASSIVE_LIB_PATH="/software/PassiveLib"
```

Patterned ground shield

Patterned ground shield

- Patterned ground shield is available in octagonal and rectangular shape



- Shape is selected with the command:

--gnd-shield-rect-geometry
--gnd-shield-oct-geometry

- Metal for patterned ground shield is chosen with the command:

--gnd-shield-metal-name=M1

Patterned ground shield

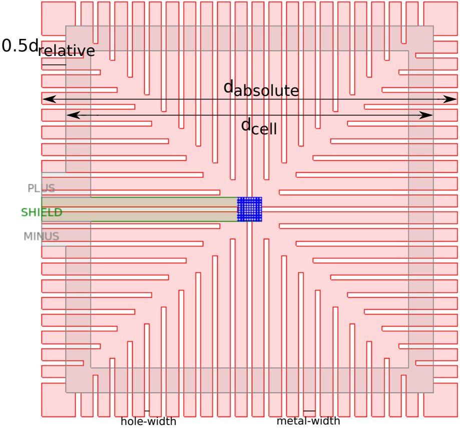
- Diameter for ground shield is set with the command:

```
--gnd-shield-diameter=10
```

- Diameter is relative in respect to the cell diameter, $d_{ABSOLUTE} = d_{RELATIVE} + d_{CELL}$
- Metal and hole widths are set with commands:

```
--gnd-shield-metal-width=5
--gnd-shield-hole-width=1
```

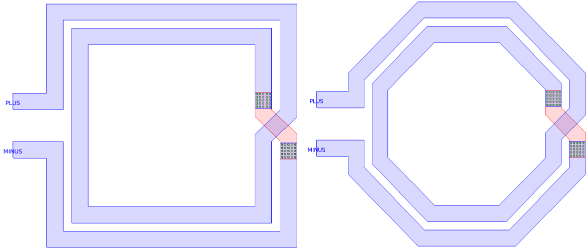
Patterned ground shield



Symmetrical inductor

Rectangular and octagonal shapes

- Symmetrical inductor can be rectangular or octagonal



set rectangular geometry

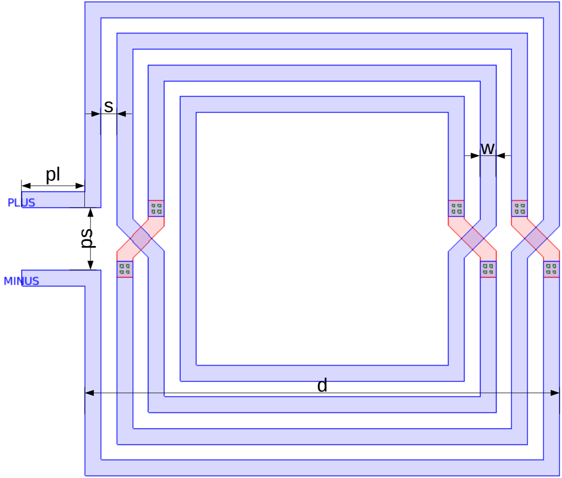
--rect-geometry

set octagonal geometry

--oct-geometry

Geometric parameters

- Symmetrical inductor:



Geometric parameters

- Set diameter d, long and short commands:

```
--diameter=100
-d 100
```

- Set metal width w, long and short commands:

```
--width=5
-w 5
```

- Set metal space s, long and short commands:

```
--space=5
-s 5
```

- Set number of turns n, long and short commands:

```
--number-of-turns=4
-n 4
```

Geometric parameters

- Set pin length *pl*, long and short commands:

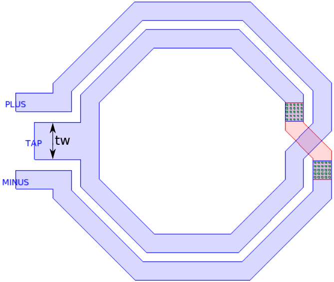
```
--pin-length=20
--pl=20
```

- Set pin space *ps*, long and short commands:

```
--pin-space=10
--ps=10
```

Geometric parameters

- Inductor can be tapped or not



- Tapping is set with the command:

--tapped=2

- Tapping is normalized to the metal width, parameter tw=tapped*w

Examples

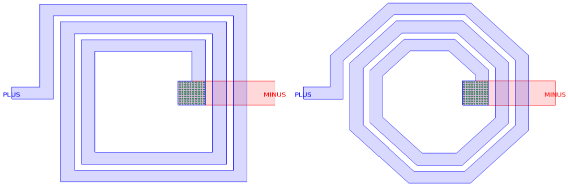
- Example symmetrical inductor:

```
PassiveLib -d 150 -w 10 -s 5 -n 2 -t inductor-symmetric
--pin-length=20 --top-metal=TM2 --cell-name=test
--gds-file-name=test.gds --oct-geometry
--gnd-shield-metal-name=M1 --gnd-shield-oct-geometry
--gnd-shield-diameter=10 --gnd-shield-metal-width=5
--gnd-shield-hole-width=2
```

Spiral inductor

Rectangular and octagonal shapes

- Spiral inductor can be rectangular or octagonal



set rectangular geometry

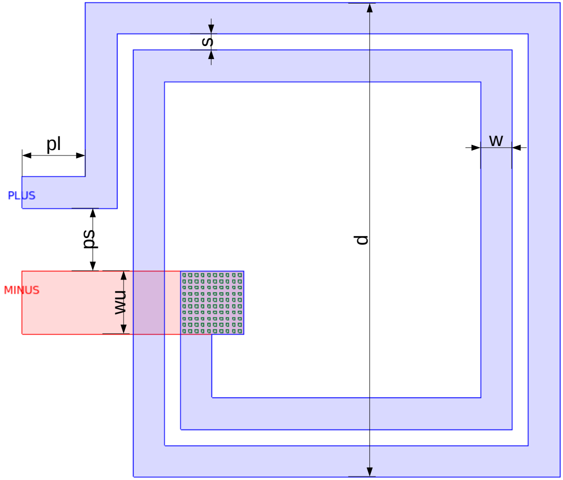
--rect-geometry

set octagonal geometry

--oct-geometry

Geometric parameters

- Spiral inductor:



Geometric parameters

- Set diameter *d*, long and short commands:

```
--diameter=100
-d 100
```

- Set metal width *w*, long and short commands:

```
--width=5
-w 5
```

- Set metal space *s*, long and short commands:

```
--space=5
-s 5
```

- Set number of turns *n*, long and short commands:

```
--number-of-turns=4
-n 4
```

Geometric parameters

- Set pin length *pl*, long and short commands:

```
--pin-length=20
--pl=20
```

- Set pin space *ps*, long and short commands:

```
--pin-space=10
--ps=10
```

- Set underpass metal width *wu*:

```
--underpass-metal-width=2
```

- Underpass metal width is normalized to the metal width, parameter wu=underpass-metal-width*w

Examples

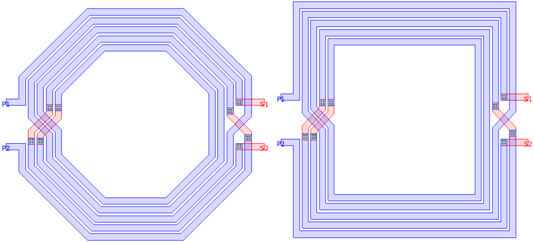
- Example spiral inductor:

```
PassiveLib -d 150 -w 10 -s 5 -n 2.5 -t inductor-spiral
--pin-length=20 --top-metal=TM2 --cell-name=test
--gds-file-name=test.gds --oct-geometry
--underpass-metal-width=2
```

Spiral transformer

Rectangular and octagonal shapes

- Spiral transformer can be rectangular or octagonal



set rectangular geometry

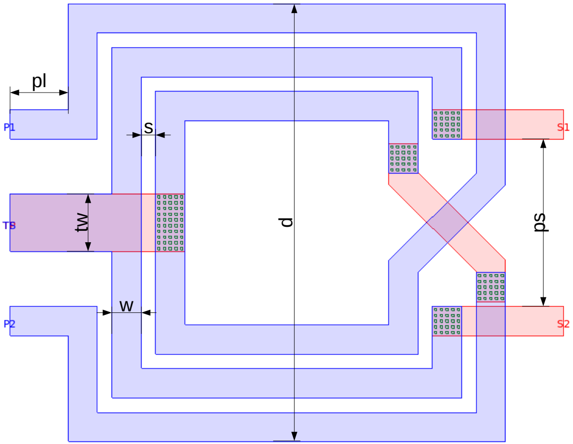
--rect-geometry

set octagonal geometry

--oct-geometry

Geometric parameters

- Spiral transformer:



Geometric parameters

- Set diameter *d*, long and short commands:

```
--diameter=100
-d 100
```

- Set metal width *w*, long and short commands:

```
--width=5
-w 5
```

- Set metal space *s*, long and short commands:

```
--space=5
-s 5
```

- Set number of primary turns *np*, long and short commands:

```
--number-of-primary-turns=2
--np=2
```

Geometric parameters

- Set number of secondary turns *ns*, long and short commands:

```
--number-of-secondary-turns=2
--ns=2
```

- Set pin length *pl*, long and short commands:

```
--pin-length=20
--pl=20
```

- Set pin space *ps*, long and short commands:

```
--pin-space=10
--ps=10
```

- Set primary and secondary tapping:

```
--tapped-primary=2
--tapped-secondary=2
```

- Tapping is normalized to the metal width, parameter tw=tapped-primary*w

Examples

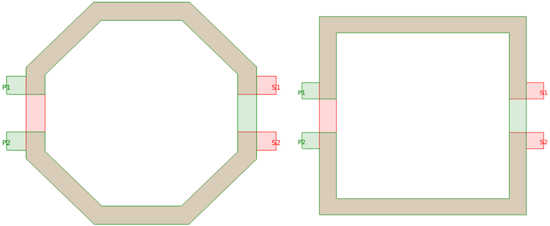
- Example spiral transformer:

```
PassiveLib -d 180 -w 5 -s 2 --np=3 --ns=2 -t
transformer-spiral --top-metal=TM2 --cell-name=test
--gds-file-name=test.gds --rect-geometry
--tapped-primary=2 --tapped-secondary=2
--gnd-shield-metal-name=M1 --gnd-shield-oct-geometry
--gnd-shield-diameter=10 --gnd-shield-metal-width=5
--gnd-shield-hole-width=2
```

Transformer1o1

Rectangular and octagonal shapes

- Transformer1o1 can be rectangular or octagonal



set rectangular geometry

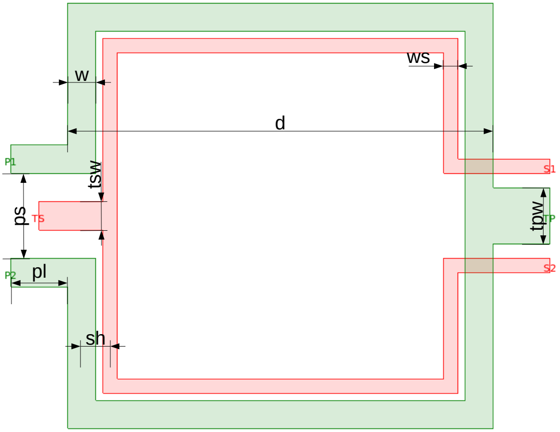
--rect-geometry

set octagonal geometry

--oct-geometry

Geometric parameters

- Transformer1o1:



Geometric parameters

- Set diameter *d*, long and short commands:

```
--diameter=100
-d 100
```

- Set primary metal width *w*, long and short commands:

```
--width=5
-w 5
```

- Set secondary metal width *ws*, long and short commands:

```
--width-secondary=5
--ws 5
```

- Set space between primary and secondary turns *sh*, long and short commands:

```
--shift-secondary=10
–sh=10
```

Geometric parameters

- Set pin length *pl*, long and short commands:

```
--pin-length=20
--pl=20
```

- Set pin space *ps*, long and short commands:

```
--pin-space=10
--ps=10
```

- Set primary and secondary tapping:

```
--tapped-primary=2
--tapped-secondary=2
```

- Tapping is normalized to the metal width, parameter tpw=tapped-primary*wp and tsw=tapped-secondary*ws

Examples

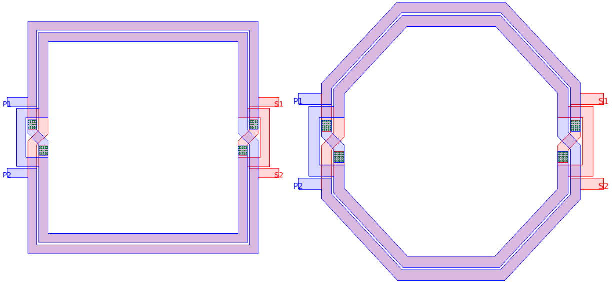
- Example transformer1o1:

```
PassiveLib -d 150 -w 10 -t transformer1o1 --sh=7 --ws=5
--pin-length=10 --top-metal=TM2 --cell-name=test
--gds-file-name=test.gds --rect-geometry
```

Transformer2o2

Rectangular and octagonal shapes

- Transformer2o2 can be rectangular or octagonal



set rectangular geometry

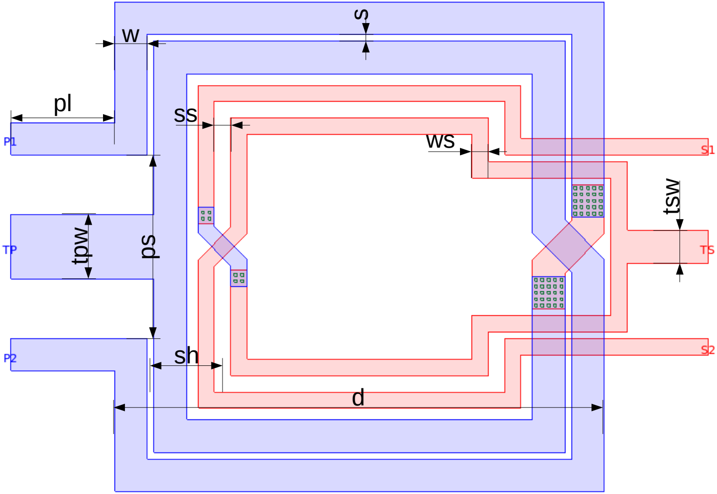
--rect-geometry

set octagonal geometry

--oct-geometry

Geometric parameters

- Transformer2o2:



Geometric parameters

- Set diameter *d*, long and short commands:

```
--diameter=100
-d 100
```

- Set primary metal width *w*, long and short commands:

```
--width=5
-w 5
```

- Set secondary metal width *ws*, long and short commands:

```
--width-secondary=5
--ws=5
```

- Set primary metal space *s*, long and short commands:

```
--space=5
-s 5
```

Geometric parameters

- Set secondary metal space *ss*, long and short commands:

```
--space-secondary=5
--ss=5
```

- Set space between primary and secondary turns *sh*, long and short commands:

```
--shift-secondary=10
--sh=10
```

- Set pin length *pl*, long and short commands:

```
--pin-length=20
--pl=20
```

- Set pin space *ps*, long and short commands:

```
--pin-space=10
--ps=10
```

Geometric parameters

- Set primary and secondary tapping:

  --tapped-primary=2
  --tapped-secondary=2

- Tapping is normalized to the metal width, parameter tpw=tapped-primary*w and tsw=tapped-secondary*ws

Examples

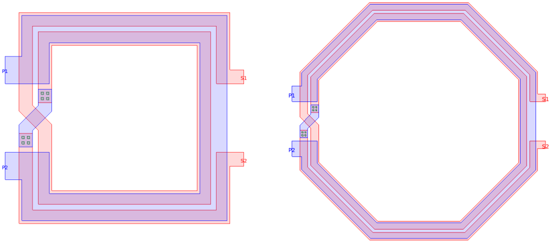
- Example transformer2o2:

```
PassiveLib -d 150 -w 10 -s 2 -t transformer2o2 --sh=7
--ss=5 --ws=7 --pin-length=10 --top-metal=TM2
--cell-name=test --gds-file-name=test.gds --rect-geometry
--tapped-primary=2 --tapped-secondary=2
```

Transformer1o2

Rectangular and octagonal shapes

- Transformer1o2 can be rectangular or octagonal



set rectangular geometry

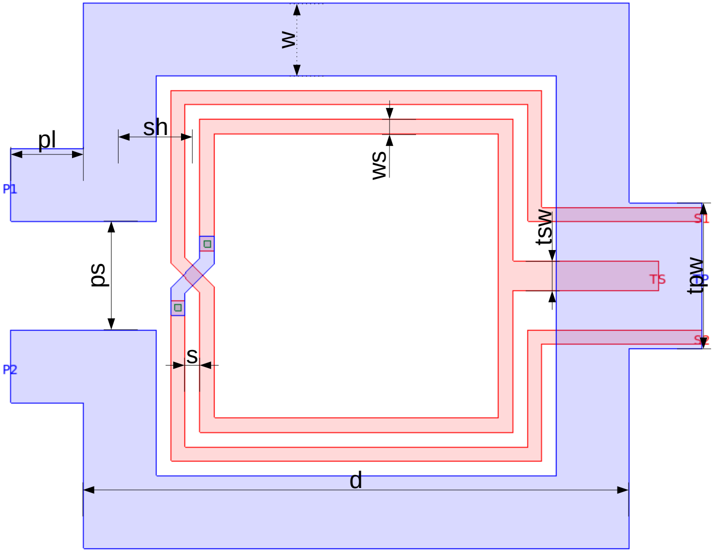
--rect-geometry

set octagonal geometry

--oct-geometry

Geometric parameters

- Transformer1o2:



Geometric parameters

- Set diameter *d*, long and short commands:

```
--diameter=100
-d 100
```

- Set primary metal width *w*, long and short commands:

```
--width=5
-w 5
```

- Set secondary metal width *ws*, long and short commands:

```
--width-secondary=5
--ws=5
```

- Set secondary metal space *s*, long and short commands:

```
--space=5
-s 5
```

Geometric parameters

- Set space between primary and secondary turns *sh*, long and short commands:

```
--shift-secondary=10
--sh=10
```

- Set pin length *pl*, long and short commands:

```
--pin-length=20
--pl=20
```

- Set pin space *ps*, long and short commands:

```
--pin-space=10
--ps=10
```

Geometric parameters

- Set primary and secondary tapping:

```
--tapped-primary=2
--tapped-secondary=2
```

- Tapping is normalized to the metal width, parameter tpw=tapped-primary*w and tsw=tapped-secondary*ws

Examples

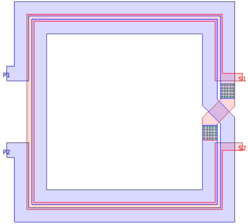
- Example transformer1o2:

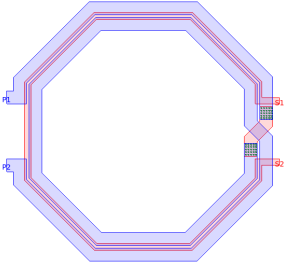
```
PassiveLib -d 150 -w 10 -s 2 -t transformer1o2 --sh=7
--ws=7 --pin-length=10 --top-metal=TM2
--cell-name=test --gds-file-name=test.gds --rect-geometry
--tapped-primary=2 --tapped-secondary=2
```

Transformer2o1

Rectangular and octagonal shapes

- Transformer2o1 can be rectangular or octagonal





set rectangular geometry

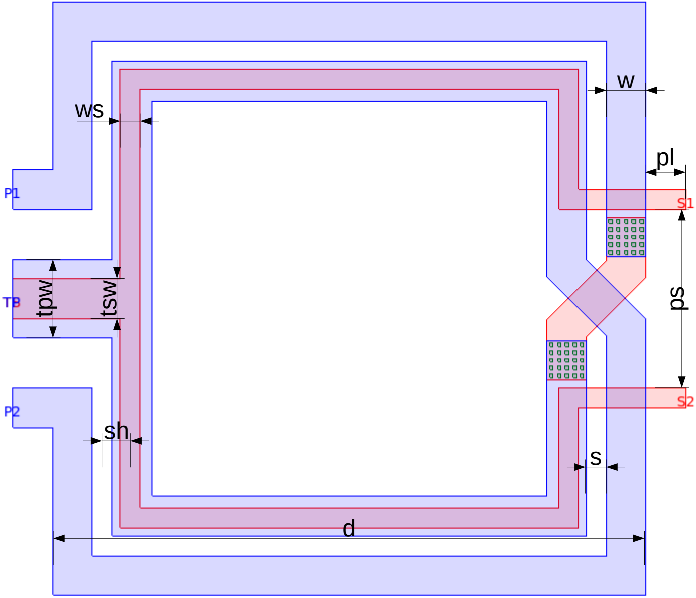
--rect-geometry

set octagonal geometry

--oct-geometry

Geometric parameters

- Transformer2o1:



Geometric parameters

- Set diameter d, long and short commands:

```
--diameter=100
-d 100
```

- Set primary metal width w, long and short commands:

```
--width=5
-w 5
```

- Set secondary metal width ws, long and short commands:

```
--width-secondary=5
--ws=5
```

- Set primary metal space s, long and short commands:

```
--space=5
-s 5
```

Geometric parameters

- Set space between primary and secondary turns *sh*, long and short commands:

```
--shift-secondary=10
--sh=10
```

- Set pin length *pl*, long and short commands:

```
--pin-length=20
--pl=20
```

- Set pin space *ps*, long and short commands:

```
--pin-space=10
--ps=10
```

Geometric parameters

- Set primary and secondary tapping:

```
--tapped-primary=2
--tapped-secondary=2
```

- Tapping is normalized to the metal width, parameter tpw=tapped-primary*w and tsw=tapped-secondary*ws

Examples

- Example transformer2o1:

```
PassiveLib -d 150 -w 10 -s 2 -t transformer2o1 --sh=7
--ws=7 --pin-length=10 --top-metal=TM2
--cell-name=test --gds-file-name=test.gds --rect-geometry
--tapped-primary=2 --tapped-secondary=2
```

Parametric model

Parametric model

- PassiveLib can sweep certain geometrical parameters and create set of gds files

```
PassiveLib -d 100:200:5 -w 2:10:1 -s 2:10:2 -n 1:5:1 -t
inductor-symmetric --pin-length=20 --top-metal=TM2
--oct-geometry --generate-spice-model
```

- Parameter *d* is swept from 100 to 200 μm with step size 5 μm
- Parameter *w* is swept from 2 to 10 μm with step size 1 μm
- Parameter *s* is swept from 2 to 10 μm with step size 2 μm
- Parameter *n* is swept from 1 to 5 with step size 1
- Option --**generate-spice-model** will prepare all needed files to generate parametric model using Cadence EMX and Cadence Modelgen
- PassiveLib creates folder gdsFile populated with gds files and folder yFile, and two additional scripts runEmx.sh and runModelgen.sh

Parametric model

- Script runEmx.sh will run Cadence EMX for every gds file in the gdsFile folder and save results in the yFile folder
- User should set environment variable **PASSIVE_LIB_EMPATH** to indicate EMX installation folder

environment variable

export PASSIVE_LIB_EMPATH="/software/emx"

- EMX control options are set with the environment variable **PASSIVE_LIB_EMOPTIONS**

environment variable

export PASSIVE_LIB_EMOPTIONS="--edge-width=1 --3d=* foundry.proc --sweep 0 20e9 --verbose=3"

- Script runModelgen.sh will run Cadence Modelgen and create spice model for the given component based on y-parameters in the yFile folder

Examples

- Example inductor-symmetric:

```
PassiveLib -d 150:200:10 -w 5:10:1 -s 5:10:1 -n 2:5:1 -t
inductor-symmetric --pin-length=20 --top-metal=TM2
--oct-geometry --gnd-shield-metal-name=M1
--gnd-shield-oct-geometry --gnd-shield-diameter=10
--gnd-shield-metal-width=5 --gnd-shield-hole-width=2
--generate-spice-model
```

- Example inductor-spiral:

```
PassiveLib -d 150:250:5 -w 5:10:1 -s 5:10:2 -n 2.5:5:0.25 -t
inductor-spiral --pin-length=20 --top-metal=TM2
--oct-geometry --underpass-metal-width=2
--generate-spice-model
```

Examples

- Example transformer-spiral:

```
PassiveLib -d 180:250:5 -w 5:10:2 -s 2:10:1 --np=3:10:1
--ns=2:10:1 -t transformer-spiral --top-metal=TM2
--rect-geometry --tapped-primary=2 --tapped-secondary=2
--gnd-shield-metal-name=M1 --gnd-shield-oct-geometry
--gnd-shield-diameter=10 --gnd-shield-metal-width=5
--gnd-shield-hole-width=2 --generate-spice-model
```

- Example transformer1o1:

```
PassiveLib -d 150:250:10 -w 5:10:1 --sh=-20:20:2
--ws=5:10:1 --pin-length=10 -t transformer1o1
--top-metal=TM2 --rect-geometry --generate-spice-model
```

Examples

- Example transformer2o2:

```
PassiveLib -d 150:250:5 -w 5:10:1 -s 2:10:1 --sh=-20:20:2
--ss=5:10:1 --ws=5:10:1 -t transformer2o2 --pin-length=10
--top-metal=TM2 --rect-geometry --tapped-primary=2
--tapped-secondary=2 --generate-spice-model
```

- Example transformer1o2:

```
PassiveLib -d 150:200:5 -w 5:10:1 -s 2:10:1 --sh=-50:50:5
--ws=5:10:1 --pin-length=10 -t transformer1o2
--top-metal=TM2 --rect-geometry --tapped-primary=2
--tapped-secondary=2 --generate-spice-model
```

Examples

- Example transformer2o1:

```
PassiveLib -d 150:250:5 -w 5:10:1 -s 2:10:1 --sh=-50:50:5
--ws=5:10:2 --pin-length=10 -t transformer2o1
--top-metal=TM2 --oct-geometry --tapped-primary=2
--tapped-secondary=2 --generate-spice-model
```

Virtuoso integration

Virtuoso integration

- Load passiveLib.ile script in the .cdsinit file

.cdsinit

```
load(strcat(getShellEnvVar("PASSIVE_LIB_PATH")
"/cds/passiveLib.ile"))
```

- Load passiveLib library in the cds.lib file

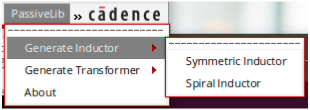
cds.lib

```
DEFINE passiveLib
$PASSIVE_LIB_PATH/cds/passiveLib
```

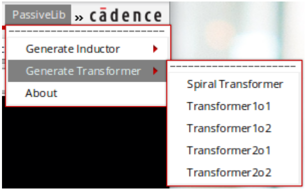
- Symbols from the provided cadence library passiveLib can be used together with generated parametric spice models to simulate custom components

Virtuoso integration

- After loading script passiveLib.ile, menu PassiveLib will be available in the layout editor
- From drop-down menu different components can be chosen



(a) inductor

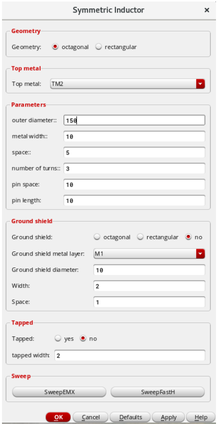


(b) transformer

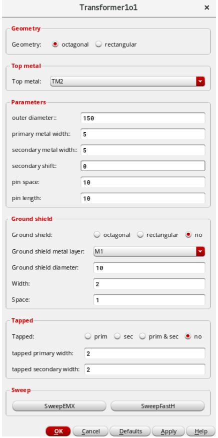
- After selecting wanted component, window from the next page will be displayed

Virtuoso integration

- Interface can be used to draw layout or to perform multidimensional sweep



(a) inductor

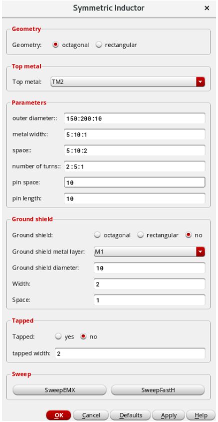


(b) transformer

- Drawing layout: specify parameters and press Apply or OK button

Virtuoso integration

- Parameters with names ending with :: are sweepable, and can be set in the format min:max:step



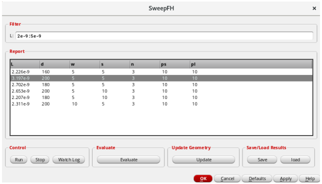
(a) inductor

(b) transformer

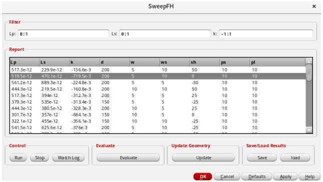
- This option is used to create multidimensional sweep over different geometries using open source FastHenry or commercial EMX solver

Virtuoso integration

- SweepFastH button from the section Sweep will open the following window:



(a) inductor



(b) transformer

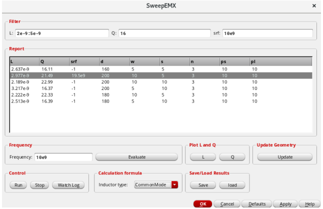
- Fast Henry can simulate large number of components in a very short time, although obtained results are calculated at DC, they can be used up to frequency 0.5*srf without significant accuracy loss
- User can use this option to quickly explore different geometries

Virtuoso integration

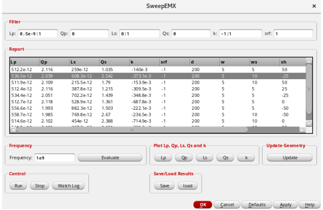
- User to do:
  - Use Run option from the Control section to run simulation
  - Use Watch Log option to see when the simulation is finished
  - If needed, terminate simulation with the Stop option
  - Once simulation is finished, calculate results using the Evaluate option
  - To show only components with required properties use the Filter section, and press again the Evaluate button
  - Format for filtering is min:max
  - Example: Lp 1e-9:1.5e-9 will show only results with Lp in range from 1nH to 1.5nH
  - After selecting component in the section Report, geometry in the window from page 72, can be updated using the option Update
  - Results can be saved for latter use by using the option Save. User has to specify a folder where results will be saved
  - If results are not saved, next run will delete old results
  - Results can be loaded by using Load option, where user needs to specify the folder which results will be loaded from

Virtuoso integration

- SweepEMX button will open the following window:



(a) inductor



(b) transformer

- EMX is full 3D electromagnetic solver and it will give results with better accuracy in comparison with FastHenry, but it will need a bit more time

Virtuoso integration

- User to do:
  - Use Run option from the Control section to run simulation
  - Once simulation is finished, specify frequency for evaluation and calculate results using the Evaluate option
  - To show only components with required properties use the Filter section, and press again Evaluate button
  - Value -1 for a self-resonance frequency (srf) means that srf is higher than maximal simulated frequency
  - simulation frequency range is specified within environment variable PASSIVE_LIB_EMOPTIONS
  - Format for filtering is min:max, or only min for Q and srf
  - Example: if we use 10 for Qp it will show all components with Qp higher than 10, if we use 10:15 it will show all components with Qp in range from 10 to 15
  - After selecting component in the section Report, user can plot L,Q and k versus frequency by using some of the options from the section Plot
  - After selecting component in the section Report, geometry in the window from page 72, can be updated using the option Update
  - Results can be saved/loaded using Save/Load options

Fast Henry

Copyright Information