



LVS Using a Customized Resistor Extractor

Aim:

To implement and test a
customized resistor extractor
for KLayout's LVS

Disclaimer:

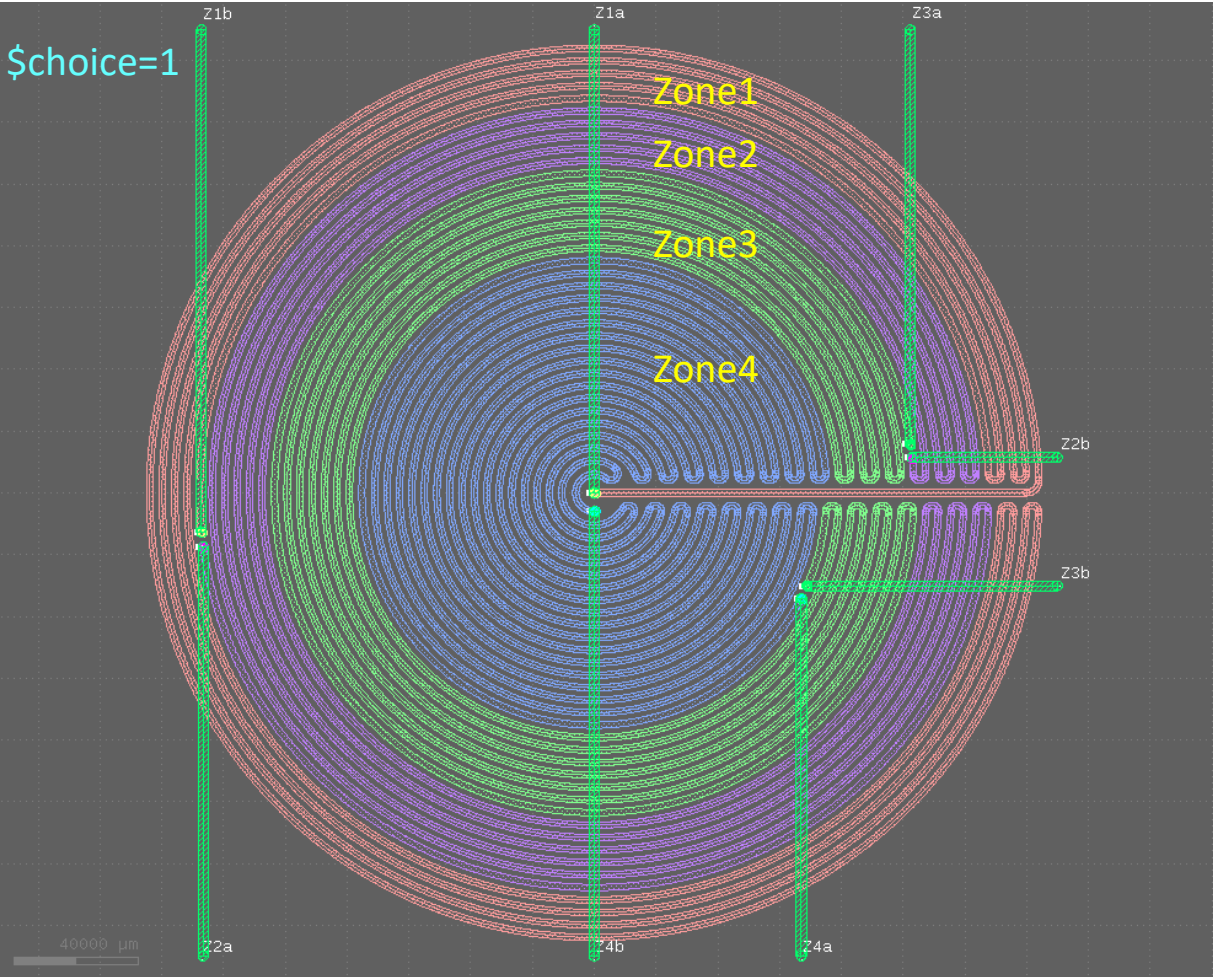
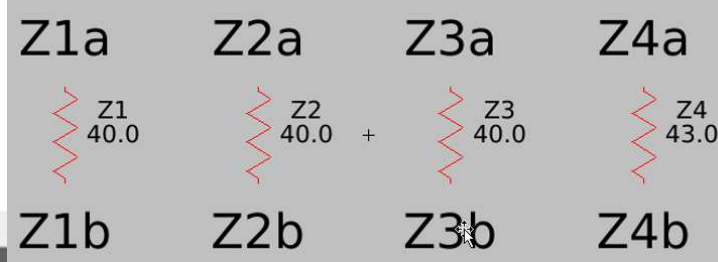
This document is for personal records only.
There is NO WARRANTY on technical
correctness.



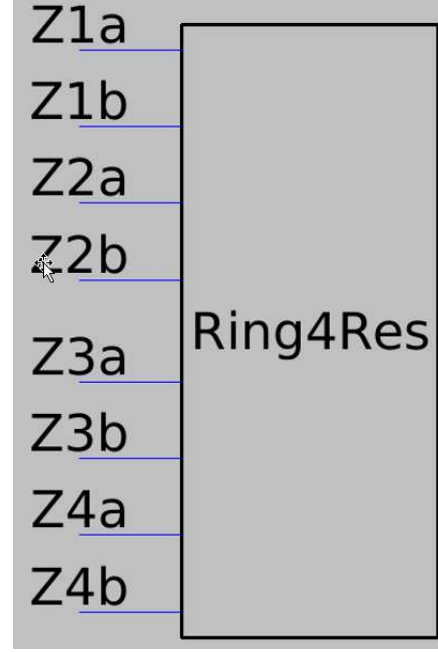
By Kazzz-S (2023-03-31)

1. The Design (BBQ hotplate with four heater zones)

- Each zone has a distinct layer set in the standard layout design.

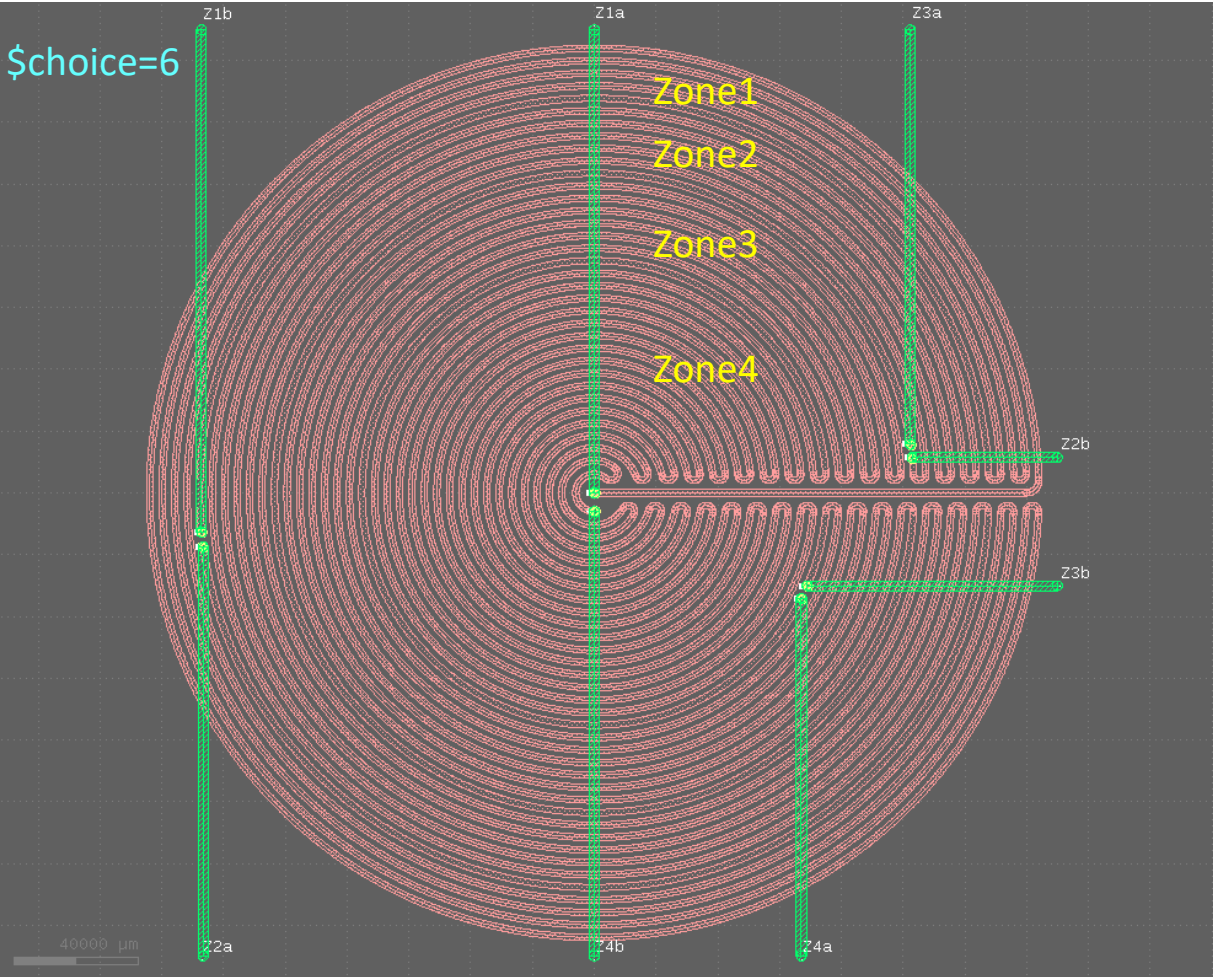
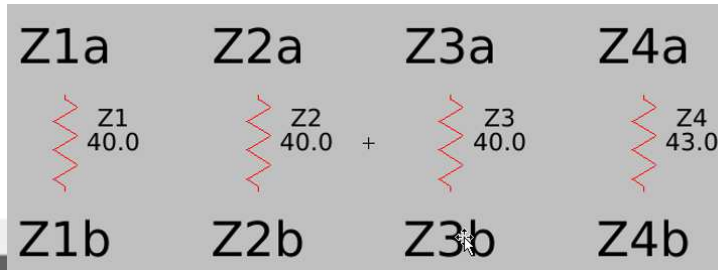


Layers	
	1002/2
	1002/402
	Zone1 3001/1
	3001/301
	3001/401
	Zone2 3002/1
	3002/301
	3002/401
	Zone3 3003/1
	3003/301
	3003/401
	Zone4 3004/1
	3004/301
	3004/401
	5002/10
	5002/0
	5002/110
	Zone1Pad 2001/1
	Zone1Cont 2011/1
	Zone1Label 2021/1
	Zone2Pad 2002/1
	Zone2Cont 2012/1
	Zone2Label 2022/1
	Zone3Pad 2003/1
	Zone3Cont 2013/1
	Zone3Label 2023/1
	Zone4Pad 2004/1
	Zone4Cont 2014/1
	Zone4Label 2024/1
	Metal1 2201/1



1. The Design (BBQ hotplate with four heater zones)

- Four zones may have a common layer set in an alternative layout design.



Layers	
	1002/2
	1002/402
	Zone1 3001/1
	3001/301
	3001/401
	Zone2 3002/1
	3002/301
	3002/401
	Zone3 3003/1
	3003/301
	3003/401
	Zone4 3004/1
	3004/301
	3004/401
	5002/10
	5002/0
	5002/110
	Zone1Pad 2001/1
	Zone1Cont 2011/1
	Zone1Label 2021/1
	Zone2Pad 2002/1
	Zone2Cont 2012/1
	Zone2Label 2022/1
	Zone3Pad 2003/1
	Zone3Cont 2013/1
	Zone3Label 2023/1
	Zone4Pad 2004/1
	Zone4Cont 2014/1
	Zone4Label 2024/1
	Metal1 2201/1



2. The Customized Resistor Extractor

```
61 class HeaterResistorExtractor < RBA::GenericDeviceExtractor
62 #
63 # Refer to $HOME/GitWork/klayout/testdata/lvs/res_combine1.lvs
64 #
65
66 def initialize(name, sheet_rho)
67   self.name = name
68   @sheet_rho = sheet_rho
69   @num_res_frgs = 0
70   @device = nil
71   @resistors = []
72 end
73
74 def setup
75   define_layer("C", "Conductor")
76   define_layer("R", "Resistor")
77   define_opt_layer("tR", 1, "Resistor")
78   register_device_class(RBA::DeviceClassResistor::new)
79 end
80
81 def get_connectivity(layout, layers)
82 # this "connectivity" forms the shape clusters that make up the device
83   conn = RBA::Connectivity::new
84   conn.connect(layers[0], layers[1]) # collect touching contacts
85   conn.connect(layers[1], layers[1]) # combine resistor shapes into one area
86   conn
87 end
88
89 def get_resistor_fragment_count()
90 # get the number of resistor fragments
91   return @num_res_frgs
92 end
93
94 def get_dev_resistance()
95 # get the device resistance
96   return @device.parameter( RBA::DeviceClassResistor::PARAM_R )
97 end
98
99 def get_resistor_array()
100 # get the resistor array
101   return @resistors
102 end
103
```

```
104 def extract_devices(layer_geometry)
105 # layer_geometry provides the input layers in the order they are defined with "define_layer"
106   conductor = layer_geometry[0]
107   resistor = layer_geometry[1]
108
109   resistor_regions = resistor.merged
110   resistor_regions.each do |r|
111     terminals = conductor.interacting(resistor)
112     if terminals.size != 2
113       @num_res_frgs += resistor_regions.count()
114       error( "Resistor shape does not touch marker border in exactly two places", r )
115     else
116       # Assume that the aspect ratio (L/W) is very large
117       #
118       #           L
119       # +-----+-----+-----+-----+-----+-----+-----+-----+-----+
120       # |                                     |                                     | W
121       # +-----+-----+-----+-----+-----+-----+-----+-----+-----+
122       #
123       # A = L*W
124       # P = 2*(L+W)
125       #
126       #           (P/2) + sqrt{ (P/2)^2 - 4A }
127       #           /
128       #   W = -----
129       #           2
130       #
131       #   L = P/2 - W
132       #
133       _a = r.area*dbu*dbu # [um^2]
134       _p = r.perimeter*dbu # [um]
135       _b = _p / 2.0
136
137       w = ( _b - Math.sqrt( _b**2 - 4.0*_a ) ) / 2.0
138       l = _b - w
139       r = @sheet_rho * l / w
140
141       @resistors << r
142
143       @device = create_device
144       @device.set_parameter( RBA::DeviceClassResistor::PARAM_R, r.round(3) );
145
146       @device.set_parameter( RBA::DeviceClassResistor::PARAM_A, _a.round(3) )
147       @device.set_parameter( RBA::DeviceClassResistor::PARAM_L, l.round(3) )
148       @device.set_parameter( RBA::DeviceClassResistor::PARAM_P, _p.round(3) )
149       @device.set_parameter( RBA::DeviceClassResistor::PARAM_W, w.round(3) )
150       define_terminal( @device, RBA::DeviceClassResistor::TERMINAL_A, 0, terminals[0] );
151       define_terminal( @device, RBA::DeviceClassResistor::TERMINAL_B, 0, terminals[1] );
152     end
153   end
154 end # class HeaterResistorExtractor
```

Estimation of L and W assuming that

- L >> W
- W is *nearly* constant everywhere

2. The Customized Resistor Extractor

```
382 #-----
383 # [5] Device extraction
384 #-----
385 # For the dummy material:
386 #   rho = 20.000 [micro-Ohm*cm]
387 #   thickness = 10.000 [um]
388 sheet_rho = 20.0E-3 # = rho * 10000[um] / thickness
389
390 ex1 = HeaterResistorExtractor::new( "Z1Heater", sheet_rho )
391 ex2 = HeaterResistorExtractor::new( "Z2Heater", sheet_rho )
392 ex3 = HeaterResistorExtractor::new( "Z3Heater", sheet_rho )
393 ex4 = HeaterResistorExtractor::new( "Z4Heater", sheet_rho )
394
395 extract_devices( ex1, { "C" => pad_zone1, "R" => rbody_zone1, "tR" => rbody_zone1 } )
396 extract_devices( ex2, { "C" => pad_zone2, "R" => rbody_zone2, "tR" => rbody_zone2 } )
397 extract_devices( ex3, { "C" => pad_zone3, "R" => rbody_zone3, "tR" => rbody_zone3 } )
398 extract_devices( ex4, { "C" => pad_zone4, "R" => rbody_zone4, "tR" => rbody_zone4 } )
399
400 resA1 = ex1.get_resistor_array()
401 resA2 = ex2.get_resistor_array()
402 resA3 = ex3.get_resistor_array()
403 resA4 = ex4.get_resistor_array()
404
405 resLayout = Hash.new()
406 CheckLayoutResistorArray( 1, ex1, resA1, resLayout )
407 CheckLayoutResistorArray( 2, ex2, resA2, resLayout )
408 CheckLayoutResistorArray( 3, ex3, resA3, resLayout )
409 CheckLayoutResistorArray( 4, ex4, resA4, resLayout )
```

Each zone has its own dedicated resistor extractor

3. Test Results of the Standard Layout Design (\$choice=1)

\$choice = 1 \$tol_rel = 3.0%

```
>>> Current design file = '/home/sekigawa/GitWork/ForumKLayout/Study004/Ring4Res_C1.oas'  
Top cell name = 'Ring4Res'  
SPICE deck file = 'Ring4Res.spi'
```

Using a Self Management Tool (SMT)

```
### Comparison succeeded with the relative tolerance of 3.000% ###
```

```
Z1: SPICE = 40.00[Ω] Layout = 39.591[Ω] Abs.diff = -0.409[Ω] Rel.diff = -1.022[%] => matched  
Z2: SPICE = 40.00[Ω] Layout = 39.545[Ω] Abs.diff = -0.455[Ω] Rel.diff = -1.137[%] => matched  
Z3: SPICE = 40.00[Ω] Layout = 39.548[Ω] Abs.diff = -0.452[Ω] Rel.diff = -1.129[%] => matched  
Z4: SPICE = 43.00[Ω] Layout = 44.278[Ω] Abs.diff = 1.278[Ω] Rel.diff = 2.973[%] => matched
```

perfect

```
### Cross reference for the matched device(s)
```

```
Dev.Name(SPICE, Layout)=( 'Z1', '$1' ): SPICE = 40.00[Ω] Layout = 39.591[Ω] Abs.diff = -0.409[Ω] Rel.diff = -1.022[%] => matched  
Dev.Name(SPICE, Layout)=( 'Z2', '$2' ): SPICE = 40.00[Ω] Layout = 39.545[Ω] Abs.diff = -0.455[Ω] Rel.diff = -1.137[%] => matched  
Dev.Name(SPICE, Layout)=( 'Z3', '$3' ): SPICE = 40.00[Ω] Layout = 39.549[Ω] Abs.diff = -0.451[Ω] Rel.diff = -1.128[%] => matched  
Dev.Name(SPICE, Layout)=( 'Z4', '$4' ): SPICE = 43.00[Ω] Layout = 44.278[Ω] Abs.diff = 1.278[Ω] Rel.diff = 2.972[%] => matched
```

perfect

```
>>> Net lists participated in the comparison
```

```
circuit Ring4Res (Z1b=Z1b, Z1a=Z1a, Z2a=Z2a, Z2b=Z2b, Z3b=Z3b, Z3a=Z3a, Z4b=Z4b, Z4a=Z4a);  
device Z1Heater $1 (A=Z1b, B=Z1a) (R=39.591, L=3997973.453, W=2019.63, A=8074428620.82, P=7999986.167);  
device Z2Heater $2 (A=Z2a, B=Z2b) (R=39.545, L=3993331.691, W=2019.628, A=8065042811.96, P=7990702.638);  
device Z3Heater $3 (A=Z3b, B=Z3a) (R=39.549, L=3993663.82, W=2019.627, A=8065710362.89, P=7991366.894);  
device Z4Heater $4 (A=Z4a, B=Z4b) (R=44.278, L=4471352.82, W=2019.662, A=9030620490.1, P=8946744.964);  
end;  
circuit RING4RES (Z1A=Z1A, Z1B=Z1B, Z2A=Z2A, Z2B=Z2B, Z3A=Z3A, Z3B=Z3B, Z4A=Z4A, Z4B=Z4B);  
device RES Z1 (A=Z1B, B=Z1A) (R=40, L=0, W=0, A=0, P=0);  
device RES Z2 (A=Z2B, B=Z2A) (R=40, L=0, W=0, A=0, P=0);  
device RES Z3 (A=Z3B, B=Z3A) (R=40, L=0, W=0, A=0, P=0);  
device RES Z4 (A=Z4B, B=Z4A) (R=43, L=0, W=0, A=0, P=0);  
end;
```

Using an instance (Xref) of the class NetlistCrossReference

3. Test Results of the Standard Layout Design (\$choice=1)

\$choice = 1 \$tol_rel = 2.5%

```
>>> Current design file = /home/sekiyawa/GitWork/ForumKLayout/Study004/Ring4Res_C1.oas'
      Top cell name = 'Ring4Res'
      SPICE deck file = 'Ring4Res.spi'
```

The SMT still works properly

```
!!! Comparison failed with the relative tolerance of 2.500% !!!
Z1: SPICE = 40.00[Ω] Layout = 39.591[Ω] Abs.diff = -0.409[Ω] Rel.diff = -1.022[%] => matched
Z2: SPICE = 40.00[Ω] Layout = 39.545[Ω] Abs.diff = -0.455[Ω] Rel.diff = -1.137[%] => matched
Z3: SPICE = 40.00[Ω] Layout = 39.548[Ω] Abs.diff = -0.452[Ω] Rel.diff = -1.129[%] => matched
Z4: SPICE = 43.00[Ω] Layout = 44.278[Ω] Abs.diff = 1.278[Ω] Rel.diff = 2.973[%] => unmatched
```

perfect

```
### Cross reference for the matched device(s)
Dev.Name(SPICE, Layout)=( 'Z1', '$1' ): SPICE = 40.00[Ω] Layout = 39.591[Ω] Abs.diff = -0.409[Ω] Rel.diff = -1.022[%] => matched
Dev.Name(SPICE, Layout)=( 'Z2', '$2' ): SPICE = 40.00[Ω] Layout = 39.545[Ω] Abs.diff = -0.455[Ω] Rel.diff = -1.137[%] => matched
Dev.Name(SPICE, Layout)=( 'Z3', '$3' ): SPICE = 40.00[Ω] Layout = 39.549[Ω] Abs.diff = -0.451[Ω] Rel.diff = -1.128[%] => matched
```

imperfect

```
>>> Net lists participated in the comparison
circuit Ring4Res (Z1b=Z1b, Z1a=Z1a, Z2a=Z2a, Z2b=Z2b, Z3b=Z3b, Z3a=Z3a, Z4b=Z4b, Z4a=Z4a);
device Z1Heater $1 (A=Z1b, B=Z1a) (R=39.591, L=3997973.453, W=2019.63, A=8074428620.82, P=7999986.167);
device Z2Heater $2 (A=Z2a, B=Z2b) (R=39.545, L=3993331.691, W=2019.628, A=8065042811.96, P=7990702.638);
device Z3Heater $3 (A=Z3b, B=Z3a) (R=39.549, L=3993663.82, W=2019.627, A=8065710362.89, P=7991366.894);
device Z4Heater $4 (A=Z4a, B=Z4b) (R=44.278, L=4471352.82, W=2019.662, A=9030620490.1, P=8946744.964);
end;
circuit RING4RES (Z1A=Z1A, Z1B=Z1B, Z2A=Z2A, Z2B=Z2B, Z3A=Z3A, Z3B=Z3B, Z4A=Z4A, Z4B=Z4B);
device RES Z1 (A=Z1B, B=Z1A) (R=40, L=0, W=0, A=0, P=0);
device RES Z2 (A=Z2B, B=Z2A) (R=40, L=0, W=0, A=0, P=0);
device RES Z3 (A=Z3B, B=Z3A) (R=40, L=0, W=0, A=0, P=0);
device RES Z4 (A=Z4B, B=Z4A) (R=43, L=0, W=0, A=0, P=0);
end;
```

Xref can only access the matched devices (correct?)

4. Test Results of the Alternative Layout Design (\$choice=6)

\$choice = 6 \$tol_rel = 3.0% /GitWork/ForumKLayout/Study004/Ring4Res_C1.oas'

```
top cell name = Ring4Res
SPICE deck file = 'Ring4Res.spi'
>>> But LVS will run for 'Ring4Res_C6.oas' in the same directory.
!!! Extractor 'ex1' found <4> candidates for the Zone1 resistor
!!! Extractor 'ex2' could not find any candidate for the Zone2 resistor
!!! Extractor 'ex3' could not find any candidate for the Zone3 resistor
!!! Extractor 'ex4' could not find any candidate for the Zone4 resistor

### Comparison succeeded with the relative tolerance of 3.000% ###
Z1: SPICE = 40.00[Ω] Layout = [39.545, 39.549, 44.278, 39.591][Ω] => multiple nets extracted (see 'Ring4Res-choice06_extracted.cir' and the cross reference)
Z2: SPICE = 40.00[Ω] Layout = Inf[Ω] Abs.diff = Inf[Ω] Rel.diff = Inf[%] => unmatched
Z3: SPICE = 40.00[Ω] Layout = Inf[Ω] Abs.diff = Inf[Ω] Rel.diff = Inf[%] => unmatched
Z4: SPICE = 43.00[Ω] Layout = Inf[Ω] Abs.diff = Inf[Ω] Rel.diff = Inf[%] => unmatched

### Cross reference for the matched device(s)
Dev.Name(SPICE, Layout)=( 'Z1', '$4' ): SPICE = 40.00[Ω] Layout = 39.591[Ω] Abs.diff = -0.409[Ω] Rel.diff = -1.022[%] => matched
Dev.Name(SPICE, Layout)=( 'Z2', '$1' ): SPICE = 40.00[Ω] Layout = 39.545[Ω] Abs.diff = -0.455[Ω] Rel.diff = -1.137[%] => matched
Dev.Name(SPICE, Layout)=( 'Z3', '$2' ): SPICE = 40.00[Ω] Layout = 39.549[Ω] Abs.diff = -0.451[Ω] Rel.diff = -1.128[%] => matched
Dev.Name(SPICE, Layout)=( 'Z4', '$3' ): SPICE = 43.00[Ω] Layout = 44.278[Ω] Abs.diff = 1.278[Ω] Rel.diff = 2.972[%] => matched

>>> Net lists participated in the comparison
circuit Ring4Res (Z4a=Z4a, Z4b=Z4b, Z2a=Z2a, Z1a=Z1a, Z2b=Z2b, Z3b=Z3b, Z3a=Z3a, Z1b=Z1b);
device Z1Heater $1 (A=Z2a, B=Z2b) (R=39.545, L=3993331.691, W=2019.628, A=8065042811.96, P=7990702.638);
device Z1Heater $2 (A=Z3b, B=Z3a) (R=39.549, L=3993663.82, W=2019.627, A=8065710362.89, P=7991366.894);
device Z1Heater $3 (A=Z4a, B=Z4b) (R=44.278, L=4471352.82, W=2019.662, A=9030620490.1, P=8946744.964);
device Z1Heater $4 (A=Z1b, B=Z1a) (R=39.591, L=3997973.453, W=2019.63, A=8074428620.82, P=7999986.167);
end;
circuit RING4RES (Z1A=Z1A, Z1B=Z1B, Z2A=Z2A, Z2B=Z2B, Z3A=Z3A, Z3B=Z3B, Z4A=Z4A, Z4B=Z4B);
device RES Z1 (A=Z1B, B=Z1A) (R=40, L=0, W=0, A=0, P=0);
device RES Z2 (A=Z2B, B=Z2A) (R=40, L=0, W=0, A=0, P=0);
device RES Z3 (A=Z3B, B=Z3A) (R=40, L=0, W=0, A=0, P=0);
device RES Z4 (A=Z4B, B=Z4A) (R=43, L=0, W=0, A=0, P=0);
end;
```

SMT is too simple to distinguish the multiple candidates

imperfect

perfect

Xref can distinguish the matched devices smartly

4. Test Results of the Alternative Layout Design (\$choice=6)

\$choice = 6 \$tol_rel = 2.5%

```
>>> Current design file = /home/sekigawa/GitWork/ForumKLayout/Study004/Ring4Res_C1.oas'
      Top cell name = 'Ring4Res'
      SPICE deck file = 'Ring4Res.spi'
>>> But LVS will run for 'Ring4Res_C6.oas' in the same directory.
!!! Extractor 'ex1' found <4> candidates for the Zone1 resistor
!!! Extractor 'ex2' could not find any candidate for the Zone2 resistor
!!! Extractor 'ex3' could not find any candidate for the Zone3 resistor
!!! Extractor 'ex4' could not find any candidate for the Zone4 resistor

!!! Comparison failed with the relative tolerance of 2.500% !!!
Z1: SPICE = 40.00[Ω] Layout = [39.545, 39.549, 44.278, 39.591][Ω] => multiple nets extracted (see 'Ring4Res-choice06_extracted.cir' and the cross reference)
Z2: SPICE = 40.00[Ω] Layout = Inf[Ω] Abs.diff = Inf[Ω] Rel.diff = Inf[%] => unmatched
Z3: SPICE = 40.00[Ω] Layout = Inf[Ω] Abs.diff = Inf[Ω] Rel.diff = Inf[%] => unmatched
Z4: SPICE = 43.00[Ω] Layout = Inf[Ω] Abs.diff = Inf[Ω] Rel.diff = Inf[%] => unmatched

### Cross reference for the matched device(s)
Dev.Name(SPICE, Layout)=( 'Z1', '$4' ): SPICE = 40.00[Ω] Layout = 39.591[Ω] Abs.diff = -0.409[Ω] Rel.diff = -1.022[%] => matched
Dev.Name(SPICE, Layout)=( 'Z2', '$1' ): SPICE = 40.00[Ω] Layout = 39.545[Ω] Abs.diff = -0.455[Ω] Rel.diff = -1.137[%] => matched
Dev.Name(SPICE, Layout)=( 'Z3', '$2' ): SPICE = 40.00[Ω] Layout = 39.549[Ω] Abs.diff = -0.451[Ω] Rel.diff = -1.128[%] => matched

>>> Net lists participated in the comparison
circuit Ring4Res (Z4a=Z4a, Z4b=Z4b, Z2a=Z2a, Z1a=Z1a, Z2b=Z2b, Z3b=Z3b, Z3a=Z3a, Z1b=Z1b);
  device Z1Heater $1 (A=Z2a, B=Z2b) (R=39.545, L=3993331.691, W=2019.628, A=8065042811.96, P=7990702.638);
  device Z1Heater $2 (A=Z3b, B=Z3a) (R=39.549, L=3993663.82, W=2019.627, A=8065710362.89, P=7991366.894);
  device Z1Heater $3 (A=Z4a, B=Z4b) (R=44.278, L=4471352.82, W=2019.662, A=9030620490.1, P=8946744.964);
  device Z1Heater $4 (A=Z1b, B=Z1a) (R=39.591, L=3997973.453, W=2019.63, A=8074428620.82, P=7999986.167);
end;
circuit RING4RES (Z1A=Z1A, Z1B=Z1B, Z2A=Z2A, Z2B=Z2B, Z3A=Z3A, Z3B=Z3B, Z4A=Z4A, Z4B=Z4B);
  device RES Z1 (A=Z1B, B=Z1A) (R=40, L=0, W=0, A=0, P=0);
  device RES Z2 (A=Z2B, B=Z2A) (R=40, L=0, W=0, A=0, P=0);
  device RES Z3 (A=Z3B, B=Z3A) (R=40, L=0, W=0, A=0, P=0);
  device RES Z4 (A=Z4B, B=Z4A) (R=43, L=0, W=0, A=0, P=0);
end;
```

SMT is too simple to distinguish the multiple candidates

imperfect

imperfect

Xref can only access the matched devices (correct?)

5. Feature Request

```
194 def ExamineCrossReference( tol_rel, dataXRef )
195   # create the device pair hash
196   devHash = Hash.new()
197   dataXRef.each_circuit_pair do |cir|
198     # puts "#{cir.status}"
199     # f = cir.first
200     # s = cir.second
201     # puts "first = #{f.name}  second = #{s.name}"
202     dataXRef.each_device_pair(cir) do |dev|
203       devF = dev.first # usually from the layout
204       devS = dev.second # usually from the reference net
205       if devF != nil and devS != nil # devF will be nil if "unmatched"
206         nameF = devF.expanded_name
207         nameS = devS.expanded_name
208         devHash[ [nameF, nameS] ] = [devF, devS]
209       end
210     end
211     # <<< Future Request >>> -----
212     #
213     # If the cause of the discrepancy is due to parameter values only, including tolerances,
214     # I want to investigate the details with a method like...
215     # NetlistCrossReference::each_unmatched_parameter_pair().
216     # ~~~~~
217     # Suppose a case of resistor, where Rext(layout)=44.278[Ω] and Rref(reference)=43.0[Ω],
218     # then Abs.diff=1.278[Ω] and Rel.diff=2.973[%].
219     # If the relative tolerance is 3.0[%], Rext and Rref match but do not if it is 2.5[%].
220     # -----
221     #
222     # dataXRef.each_unmatched_parameter_pair(dev) do |par|
223     #   ~~~~~
224     #   if par != nil
225     #     id = par.id # like RBA::DeviceClassResistor::PARAM_R
226     #     valF = par.first # first unmatched parameter value like "44.278[Ω]"
227     #     valS = par.second # second unmatched parameter value like "43.0[Ω]"
228     #     CheckWhyThisParameterUnmatched( dev, id, tol_rel, valF, valS )
229     #   end
230     # end
231 end
232 end
```

Is this method a possibility?



All Files

Ring4Res.zip

