

Perform Layout & Circuit Modifications Using RedHawk-SC

RedHawk-SC Modular Training Series

Version : 2020_R3



/ Pre-requisites for the training

No	Training Program	Expectations – Must Know
1	RedHawk-SC Modular Training <ul style="list-style-type: none">Chapter 01 : Introduction_to_SeaScape	<ul style="list-style-type: none">Reading in input data, performing data integrity checks and creating base views

Agenda

- Introduction
- Circuit Elements
- ModifiedExtractView
- Modification functions

Introduction

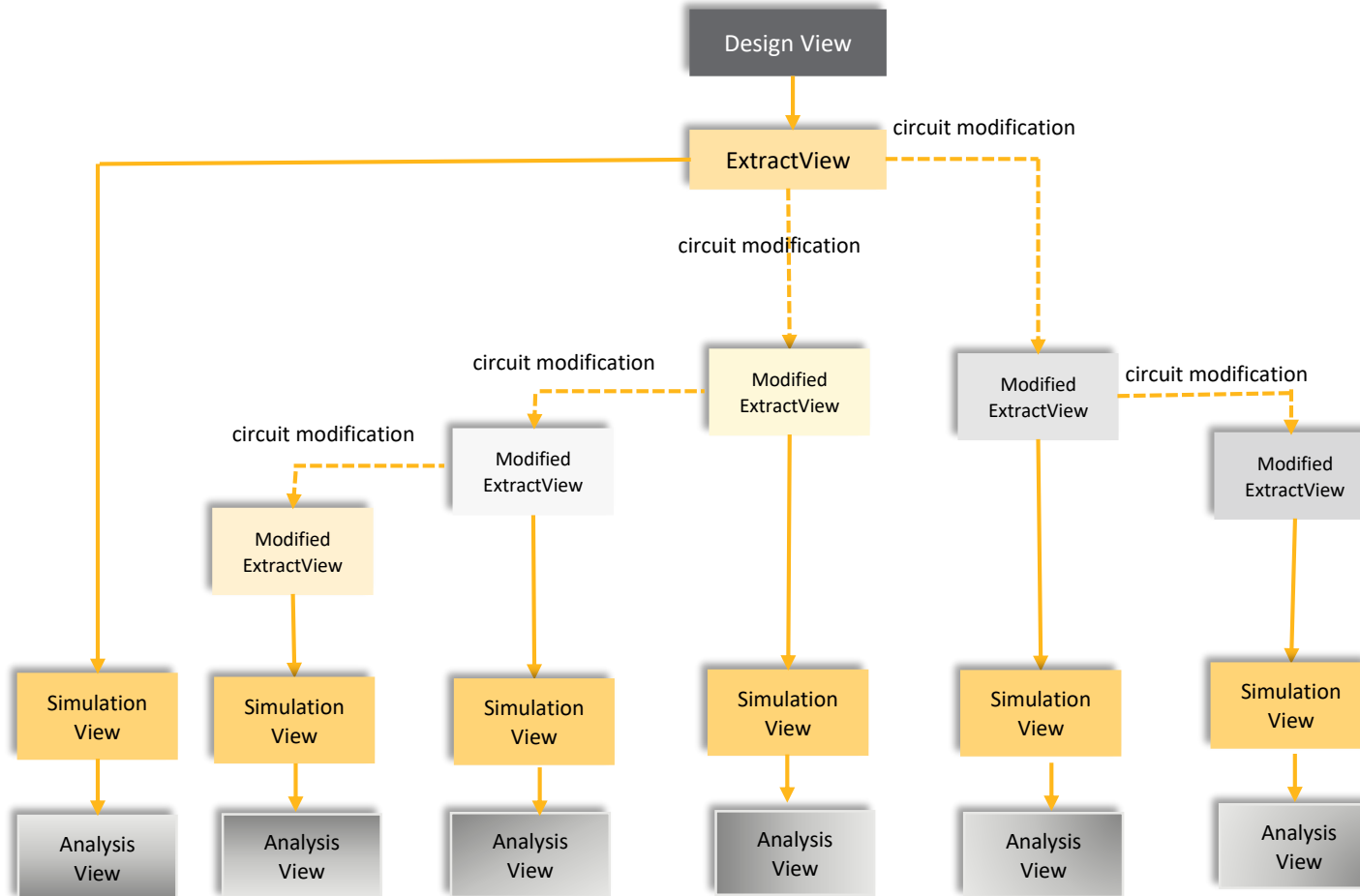


Introduction

- **RedHawk-SC** provides the infrastructure to query the circuit elements in the design and modify them.
- Compared to Layout Modification's, Modification's done at circuit level are very quick and easy to do.
- It gives flexibility to the user for experimenting without changing actual layout.

Introduction

- Figure shows Circuit Modification's flow in RedHawk-SC. User can perform multiple circuit modification's and create ModifiedExtractView's from original ExtractView. ModifiedExtractView can be chained together to apply incremental changes.

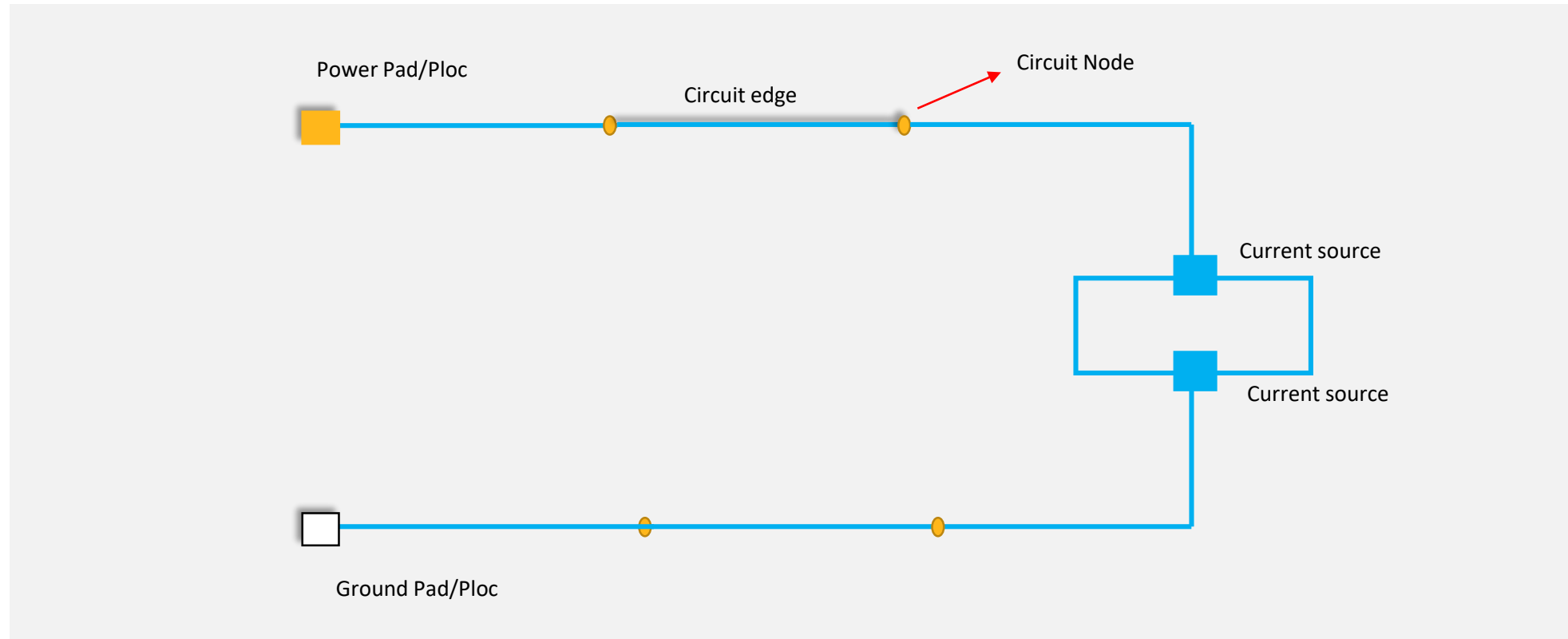


ModifiedExtractView can be passed to SimulationView to perform multiple Static/Dynamic Analysis experiments.

Circuit Elements

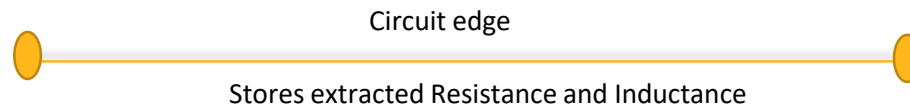
Circuit Elements

Circuit contains extracted parasitic resistors/capacitors/inductors (aka R/L/C) elements from power grid combined with power source and current sources (which represent the actual devices)



/ Circuit Elements

Circuit Edge stores extracted PG Grid resistance and inductance value



Circuit Node stores extracted PG Capacitance



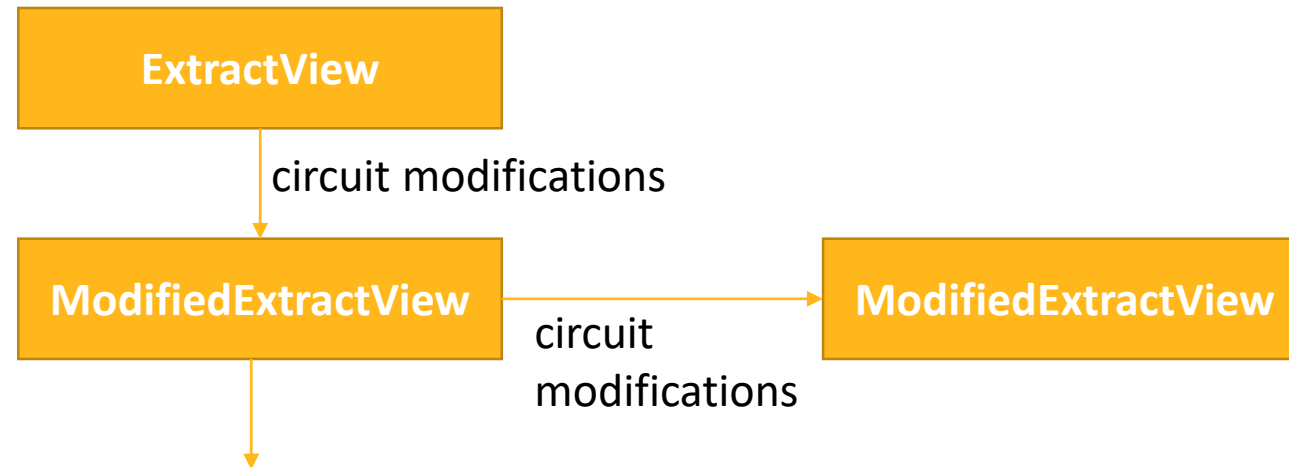
ModifiedExtractView

ModifiedExtractView

User can perform incremental PG Grid changes at circuit level on the original ExtractView and create a new **ModifiedExtractView** using create_modified_extract_view command.

ModifiedExtractView is commonly abbreviated as MEV. Following is the command to generate an MEV

Incremental circuit modifications can be done on **ModifiedExtractView** and new **ModifiedExtractView** can be created as shown below



ModifiedExtractView

create_modified_extract_view function is used to create ModifiedExtractView

```
>>> help(db.create_modified_extract_view)
create_modified_extract_view(extract_view, eco_function=None, probes=None, tag='mev', options=None, exceptions_as_errors=True)
    Create an ModifiedExtractView object based on provided circuit eco or user probes.

ARGUMENTS
    extract_view : ExtractView object (type=ExtractView, required=True)
    eco_function : Function object used to modify the provided circuit (type=object, default_value=None)
    probes : ChunkedData that stores probes (type=object, default_behavior='No custom probes will be added', constraint="one of list or ChunkedData or GpDelayedObject or MapReduce")
    tag : Name of the view (type=str, default_value='mev')
    options : View creation options, typically from get_default_options() (type=object, default_value=None)
    exceptions_as_errors : Turn exceptions inside circuit_based_rollup function to errors (type=bool, default_value=True)

RETURN VALUE
ModifiedExtractView object
```

Modification Functions

Modification Functions

RedHawk-SC provides infrastructure to perform following Circuit Modifications

- Modifying R,L,C parameters
- Deleting extracted grid parasitics
- Deleting bumps
- Adding probes

User can create circuit custom modification functions or use the existing public API's. These functions must be passed to **eco_function** inside create_modified_extract view as shown below

```
>>> help(db.create_modified_extract_view)
create_modified_extract_view(extract_view, eco_function=None, probes=None, tag='mev', options=None, exceptions_as_errors=True)
    Create an ModifiedExtractView object based on provided circuit eco or user probes.

ARGUMENTS
    extract_view : ExtractView object (type=ExtractView, required=True)
    eco_function : Function object used to modify the provided circuit (type=object, default_value=None)
    probes : ChunkedData that stores probes (type=object, default_behavior='No custom probes will be added', constraint="one of list or ChunkedData or GpDelayedObject or MapReduce")
    tag : Name of the view (type=str, default_value='mev')
    options : View creation options, typically from get_default_options() (type=object, default_value=None)
    exceptions_as_errors : Turn exceptions inside circuit_based_rollup function to errors (type=bool, default_value=True)

RETURN VALUE
ModifiedExtractView object
```

/ In-built Circuit Modification Functions

- Below are the existing customizable API's available for circuit modifications.
 1. Sparsifying bumps
 2. Deleting Specific bumps
 3. Changing Interconnect R,C values (R Multiplier and C Multiplier)
 4. Deleting extracted grid parasitics of a layer
 5. Adding Probes

All the above functions are part of **ckteco** module

/ In-built Circuit Modification Functions

```
>>> help(ckteco)
FUNCTIONS
change_values_func(dv, r_multiplier=None, c_multiplier=None) Changing Interconnect R,C parameters
    Returns a function for changing circuit element values per layer.

create_probes_on_grid(dv, layers=None, nets=None, pitch_x=100, pitch_y=100, offset=1)
    Returns a function for creating probes on a regular grid. Adding Probes on Grid

delete_bumps_func(ev, bumps) Deleting Specific Bumps
    Delete specific bumps.

delete_layer_func(dv, layers, max_search_distance=100)
    Delete all edges on given layers and reconnect. Deleting Circuit Parasitics

grid_augment_func(ev, augment_instances, layers, net=None, augment_distance_x=2.0, augment_distance_y=0.0, r_multiplier=0.5)
    Change grid resistance values around specific instances.

>>> sparsify_bumps_func(ev, delete_ratio, nets=None)
    Delete some of the bumps. Reducing the bumps in the design based on user specified ratio
```


/Sparsifying bumps function

- This in-built modification function allows the user to reduce the number of bumps in the design. User needs to provide the % of bumps that needs to be deleted in the design
- Delete algorithm tries to delete the bumps uniformly.

```
import ckteco

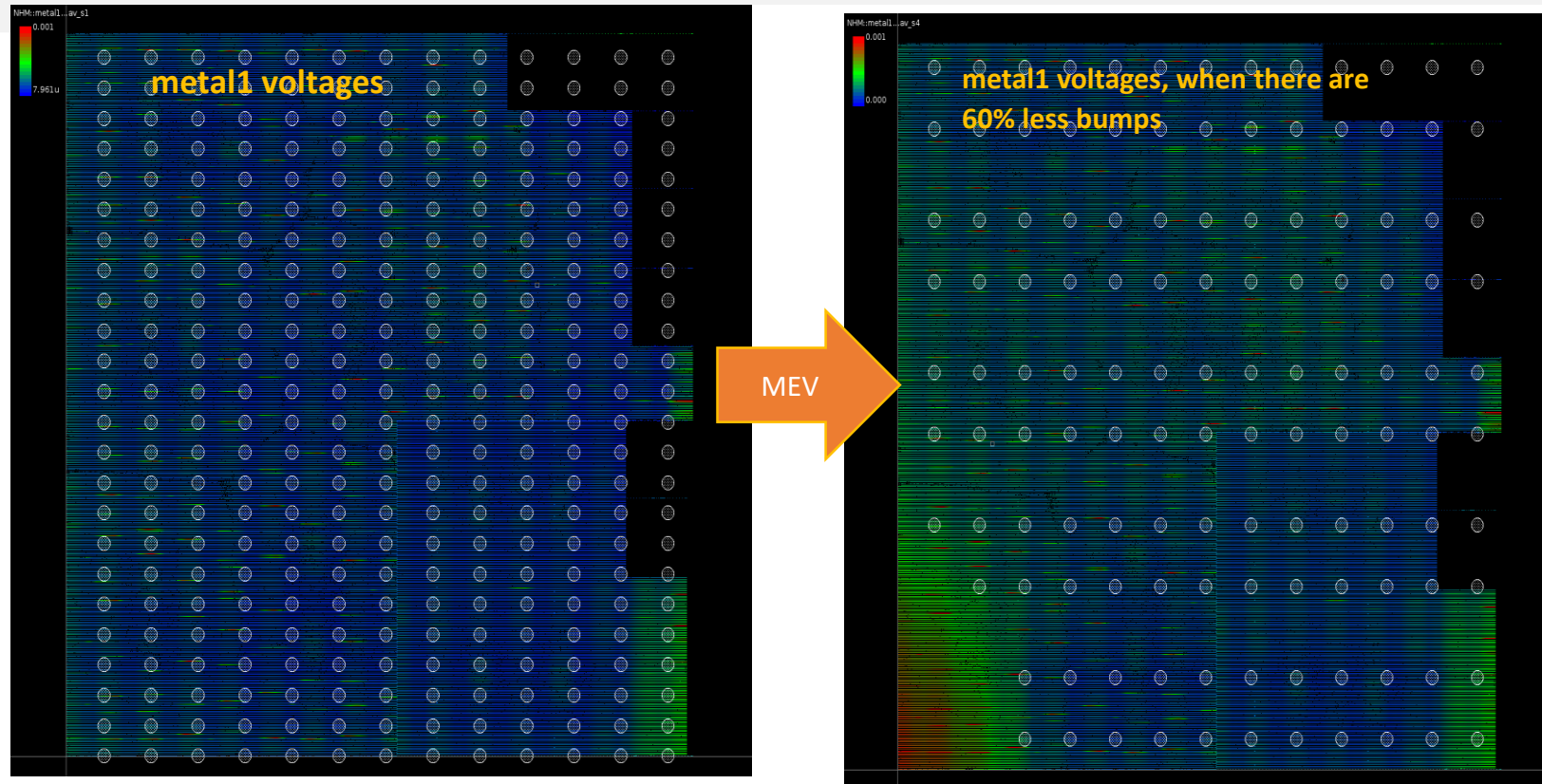
mev=db.create_modified_extract_view(ev,eco_function=ckteco.sparsify_bumps_func(ev=ev,delete_ratio=0.6),
tag='mev')
```

/Sparsifying bumps function

File : `../scripts/circuit_modifications.py`

```
import ckteco
```

```
mev=db.create_modified_extract_view(ev,eco_function=ckteco.sparsify_bumps_func(ev=ev,delete_ratio=0.6),  
tag='mev')
```



/Deleting Specific bumps function

- This in-built modification function allows the user to provide the list of existing bumps to be deleted in the design.

```
import ckteco

mev=db.create_modified_extract_view(ev,eco_function=ckteco.delete_bumps_func(ev=ev,['VSS'+str(n) for n
in range(90,210)])), tag='mev')

#delete all bumps name VSS_90, VSS_91, .... VSS_210
```

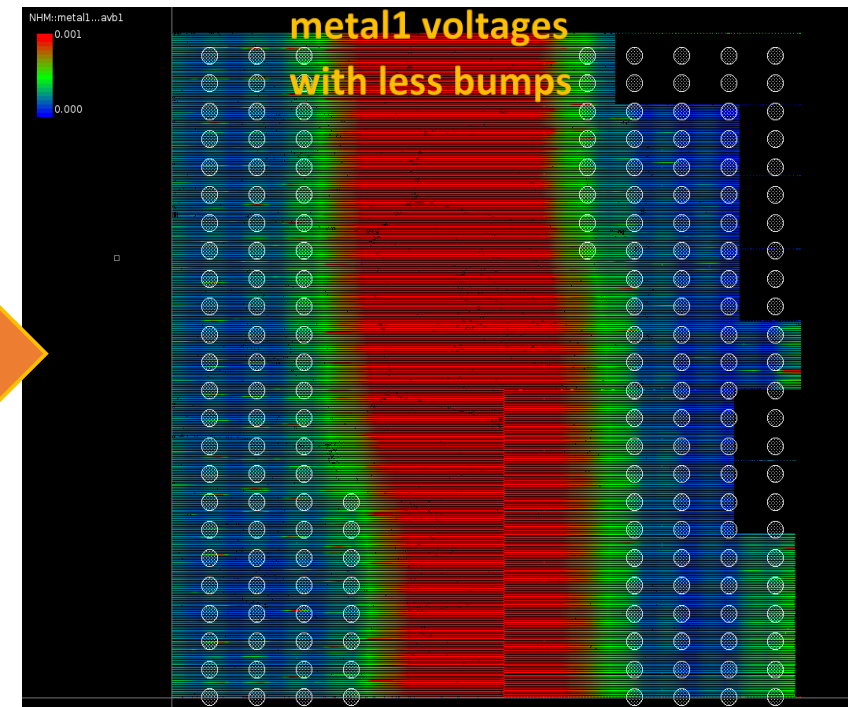
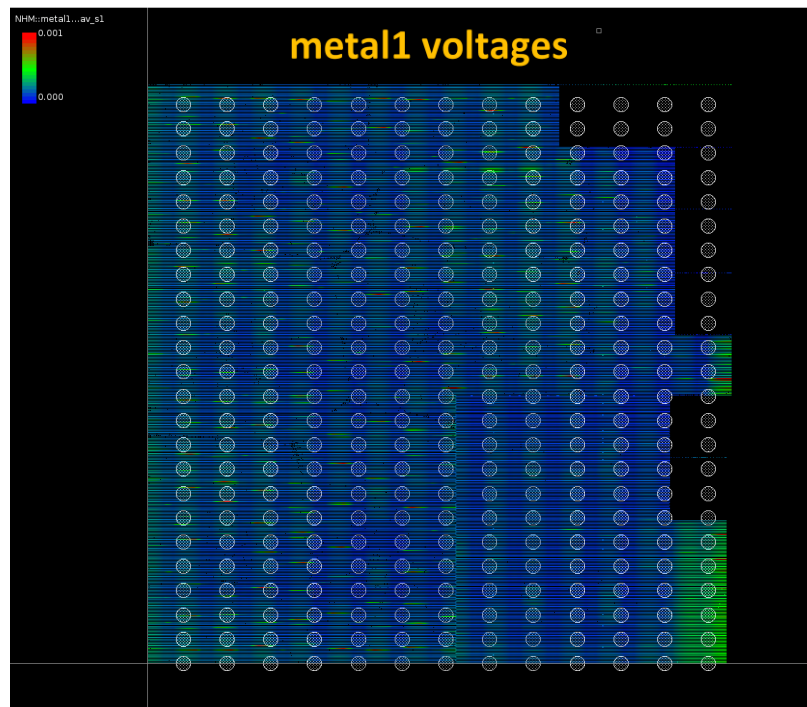
/Deleting Specific bumps function

File : `../scripts/circuit_modifications.py`

```
import ckteco
```

```
mev=db.create_modified_extract_view(ev,eco_function=ckteco.delete_bumps_func(ev=ev, ['VSS'+str(n) for n in range(90,210)])), tag='mev')
```

```
#delete all bumps name VSS_90, VSS_91, .... VSS_210
```



Changing interconnect R, C values (R-Multiplier and C-Multiplier)

- This in-built modification function allows users to scale the Resistance, Capacitance values of any layer in the design.
- User must provide dict of layer and its scaling value for Resistance and Capacitance to eco_function

```
import ckteco
r_multiplier = {Layer('metal10'):10, Layer('metal11'):10, Layer('metal12'):10}

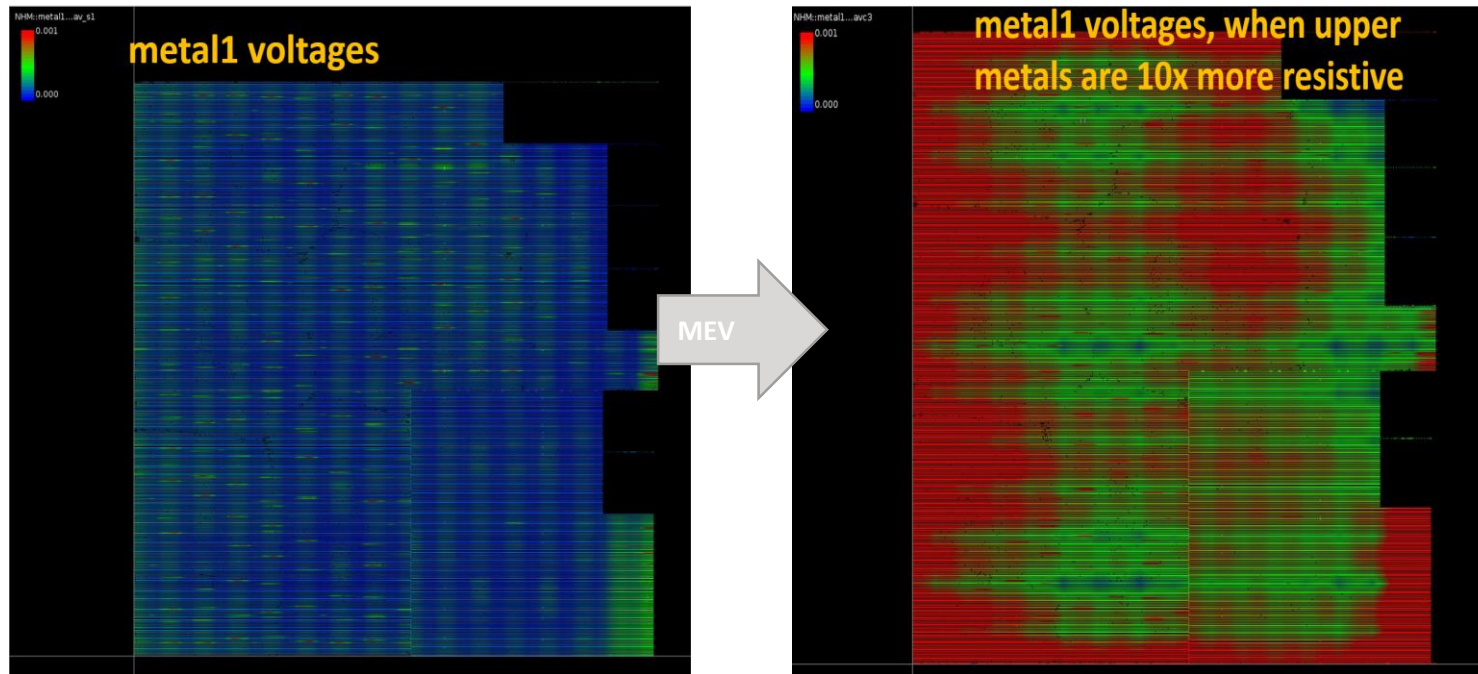
mev=db.create_modified_extract_view(ev,eco_function=ckteco.change_values_func(dv=dv,
r_multiplier=r_multiplier), tag='mev')
```

Multiplying resistors on metal10, metal11 and metal12 by 10

File : `../scripts/circuit_modifications.py`

```
import ckteco
r_multiplier = {Layer('metal10'):10, Layer('metal11'):10, Layer('metal12'):10}

mev=db.create_modified_extract_view(ev,eco_function=ckteco.change_values_func(dv=dv,
r_multiplier=r_multiplier), tag='mev')
```



Increasing metal10, metal11, metal12 capacitance by 15%



File : `../scripts/circuit_modifications.py`

```
import ckteco
c_multiplier = {Layer('metal10'):15, Layer('metal11'):15, Layer('metal12'):15}

mev=db.create_modified_extract_view(ev,eco_function=ckteco.change_values_func(dv=dv,
c_multiplier=c_multiplier), tag='mev')
```



15% Increase in the PG cap on Metal 10, Metal 12 and Metal 13 is yielding in 1.5% increase in Total Cdie at operating frequency of 125 MHz

Measure Window					
signal	0.12509Gx 1s	cur2 x 1s	deltax 1s	rising	
 cdie_freq	4.6847n				
 cdie_freq	4.6156n				

/Deleting Extracted gird parasitics of a layer

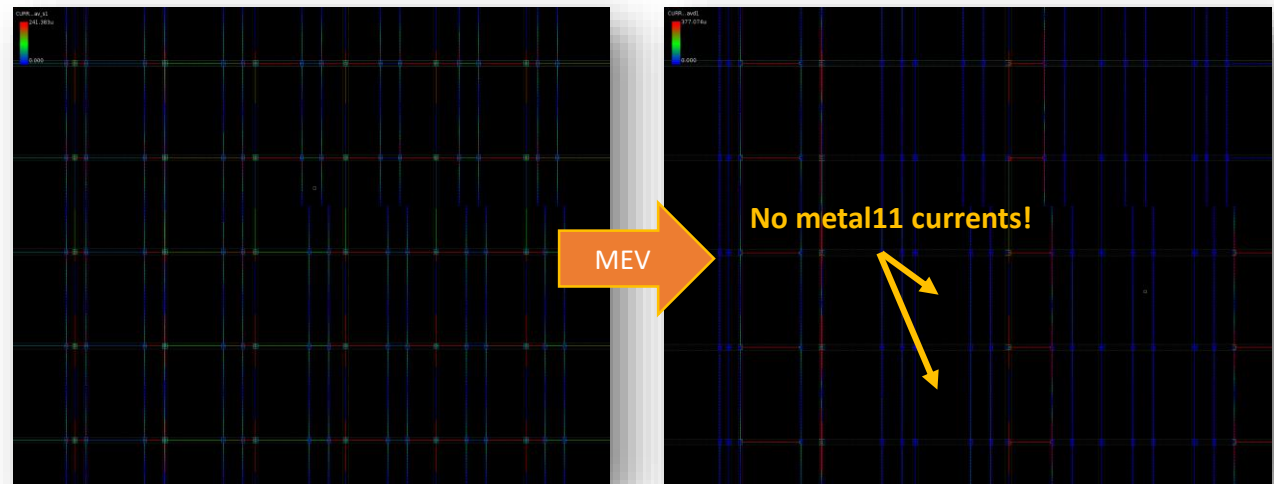
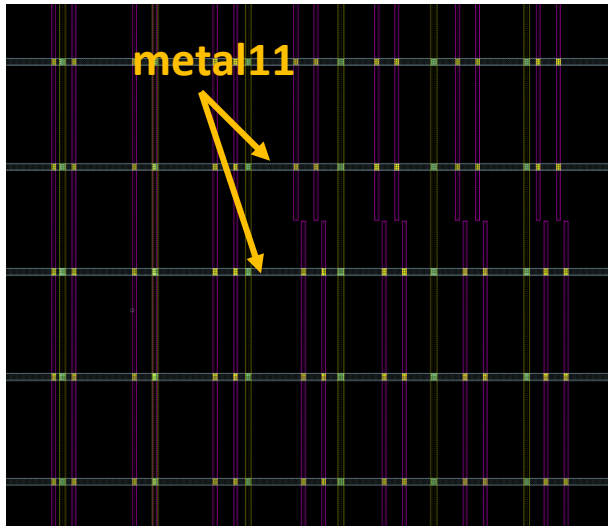
- This in-built modification function allows user to delete the circuit parasitics of any layer. User must provide list of layers to be deleted to the eco_function

```
import ckteco  
mev=db.create_modified_extract_view(ev,eco_function=ckteco.delete_layer_func(dv=dv,layer=[Layer('metal11'  
'')]),tag='mev')
```


/Deleting complete metal1 grid parasitics

File : `../scripts/circuit_modifications.py`

```
import ckteco  
  
mev=db.create_modified_extract_view(ev,eco_function=ckteco.delete_layer_func(dv=dv,layer=[Layer('metal11')]),tag='mev')
```



/Adding Probes

- Probes can be added through ModifiedExtractView or using ckteco function.
- ModifiedExtractView allows user to add probes on metal in any custom fashion as required by the design style. User can give the exact locations where probes needs to be added. However, when probes need to be placed at certain pitch on metal in-built modification function can be used.

/ Adding probes at x, y pitch

- Below is the in-built modification function to add probes on metal layers at user specified x pitch and y pitch

```
import ckteco  
  
mev=db.create_modified_extract_view(ev,eco_function=ckteco.create_probes_on_grid(dv=dv,layer=[Layer('metall')],net=[Net('V  
DD')],pitch_x=10, pitch_y=10, offset=2),tag='mev')
```

/ Adding user specified probes

- probes argument of ModifiedExtractView allows adding user defined custom probes.
- User can add probe on any metal layer by providing the x,y location and corresponding net name as shown below

```
user_probes = [{'net':Net('VDD'),'name':'Probe1','layer':Layer('metal1'),'coord':RealCoord(189,46)},  
{'net':Net('VDD'),'name':'Probe2','layer':Layer('metal1'),'coord':RealCoord(43,297)}]  
  
ev =db.create_extract_view(dv1,tech_view=nv,temperature=105,options=options,tag =  
'ev',enable_via_clustering='off',calculate_spr=True)  
mev =db.create_modified_extract_view(ev, tag='mev',probes=user_probes,options=options)
```

/Writing Custom modification function

- Users can write custom functions to modify R,L,C values.
- `eco_function` inside `create_modified_extract_view` allows user to operate only on circuits. Hence user must first loop through circuits in design and then query the edges/nodes in the modification function.
- Every edge has extracted resistance, inductance and connected node capacitances which can be queried and modified.
- Compared to in-built function for `R_multiplier` and `C_multiplier`, here we are querying the edges and nodes from circuit which allows us to query the edge properties like edge and node coordinates, edge resistance, node capacitance. This allows user to do circuit modifications applying conditions based on original queried values.

 **Ansys**

