RedHawk-SC Chip Power Model (CPM) Training

RedHawk-SC Modular Training Series

Release 2020 R3

Ansys

Info for Attendees

- This training session is for 2 hours with 10 mins Q&A session at the end
- To ask questions, please use the Q&A window and one of the panelists will answer it.
- For listening to audio , please use Audio broadcast option

Tutor and Panelists

- **Tutors**
  - Ramesh Agarwal – Lead Product Specialist

- **Panelists**
  - Siddalingesh Tenginakai – Lead Product Specialist
  - Eldo N Baby – Lead Product Specialist
  - Sankar Ramachandran – Director Product Specialist

Prerequisites for the training

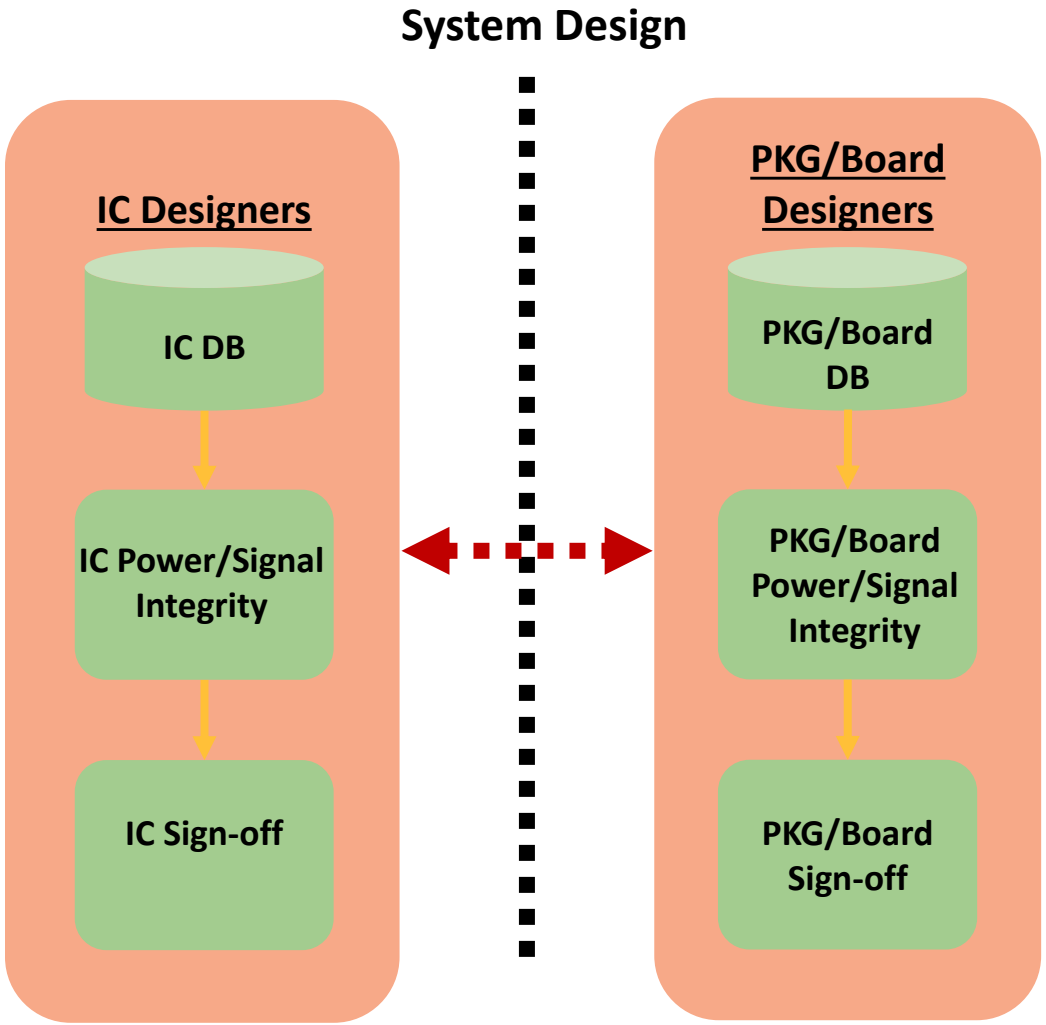
No	Training Program	Expectations – Must Know
1	RedHawk-SC Modular Training Chapter 01 : Introduction_to_SeaScape	• Reading in input data, performing data integrity checks and creating base views
2	RedHawk-SC Modular Training Chapter 04 : Dynamic_Vectorless_Analysis	• Vectorless Dynamic Analysis

Training Agenda

- Introduction to CPM
- What's in a CPM ?
- CPM Use
- Getting Familiar with Labs
- CPM Generation
- Additional CPM Configuration Options

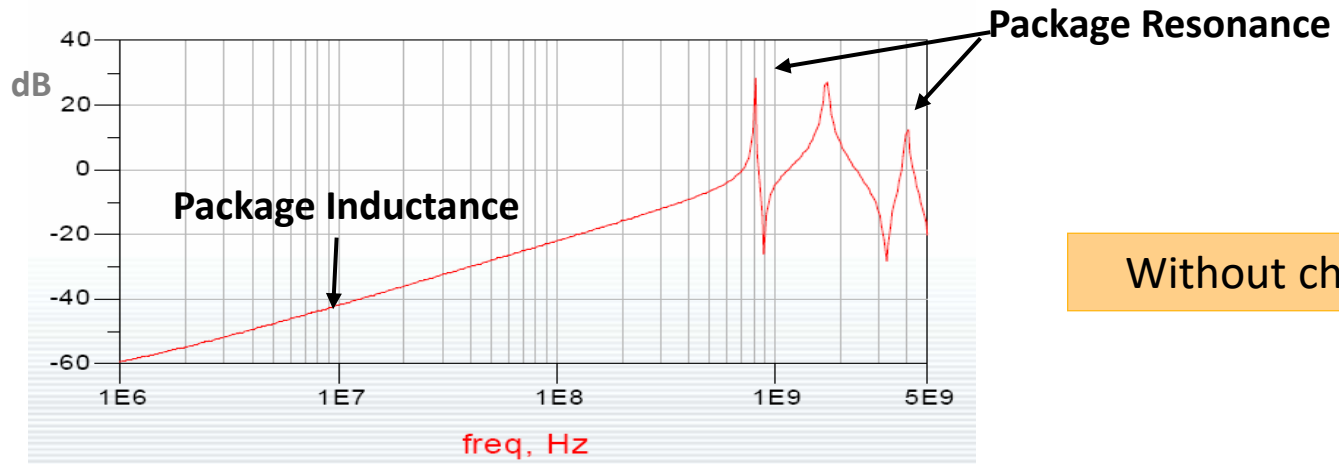
Introduction to CPM

IC-Package-PCB Co-Design Challenges

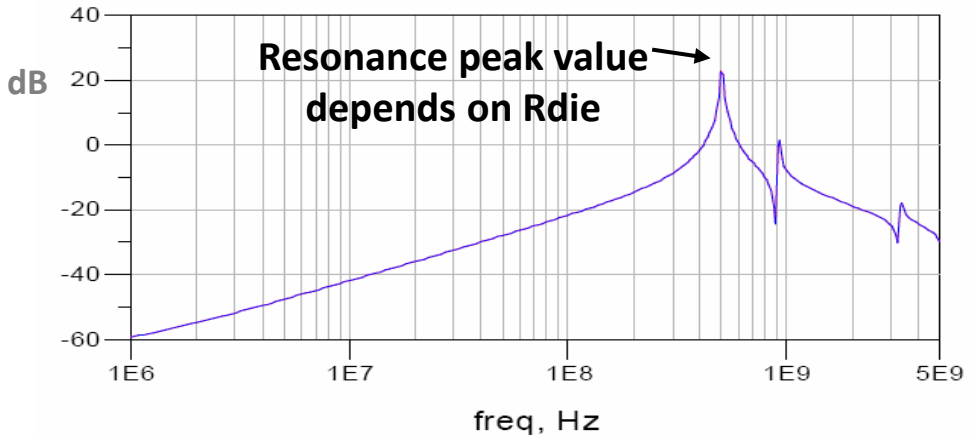


- Longer System Design Cycle
  - Chip is the source of noise
  - Lack of noise budgeting at board & package
  - Possible die-package resonance
  - Package re-spin
- Higher PKG / Board Cost
  - Over-design
  - Excessive decap

Impact of On-Die Parasitics : PDN Input Impedance vs. Freq



Without chip power model

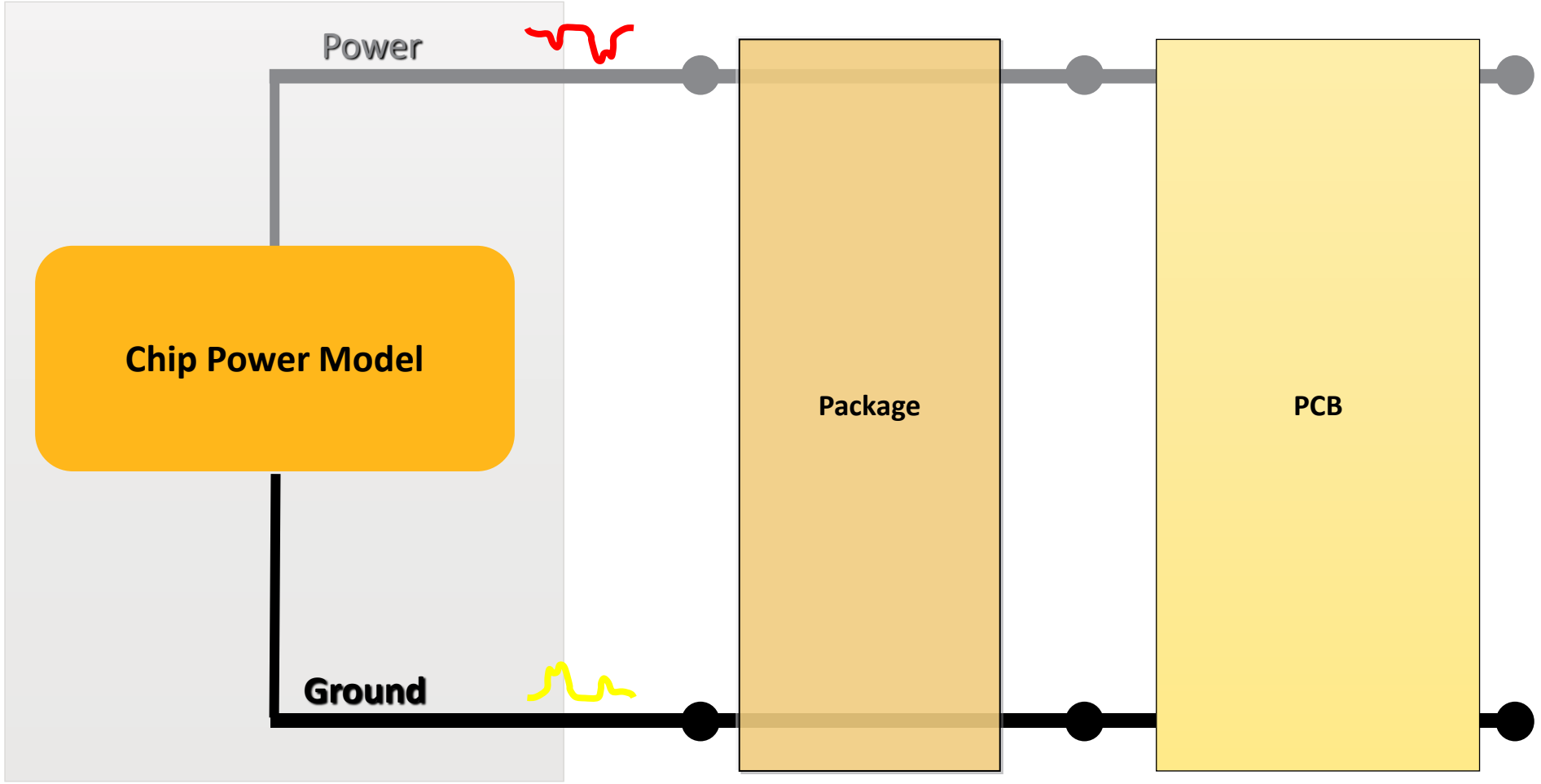


With chip power model

The resonance peak is proportional to a factor called Q where $Q = \frac{\sqrt{(L/C)}}{R}$

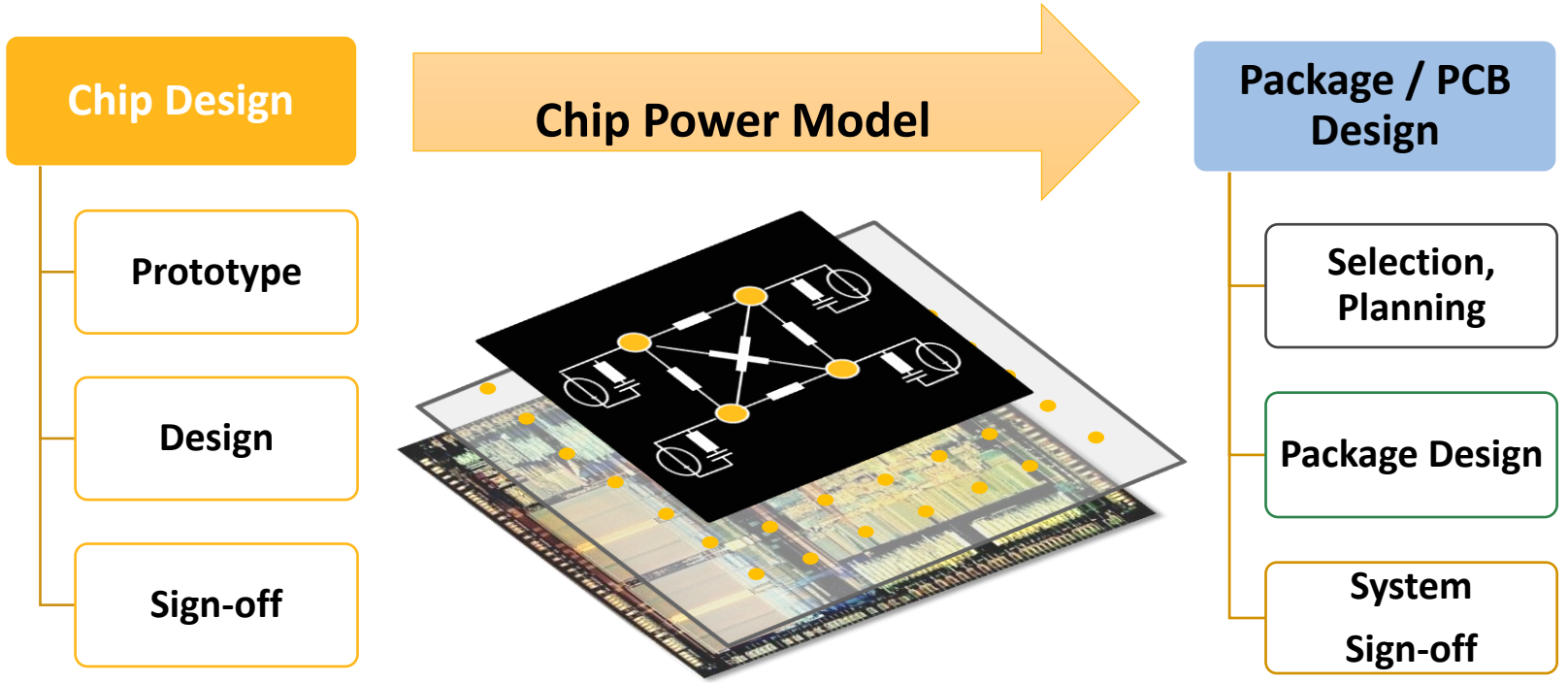
Q = 1/R * sqrt(L/C)

Chip Power Model Concept



Chip is the source of noise

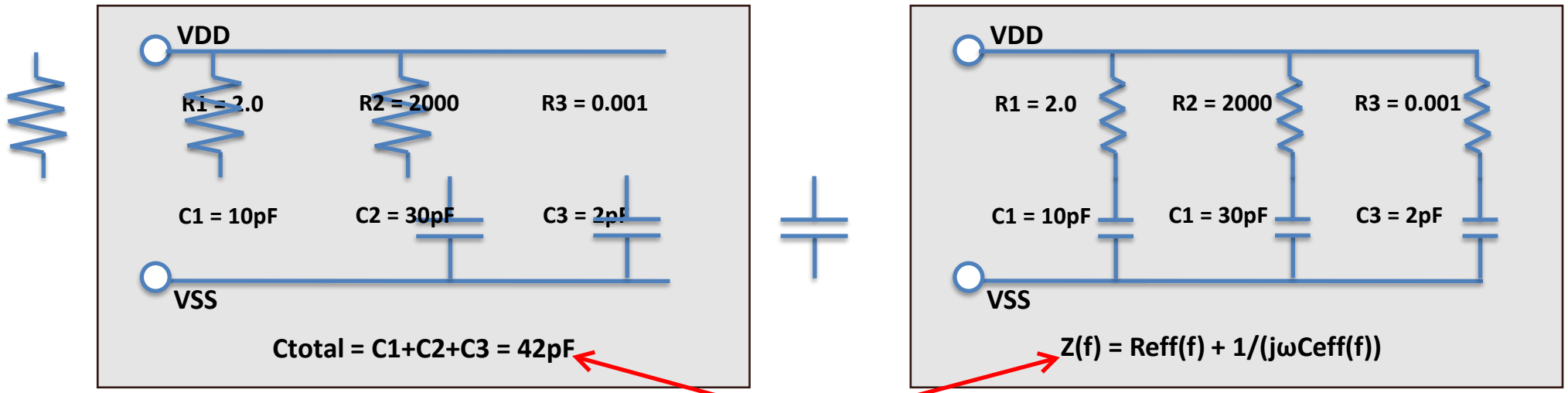
CPS Convergence Using CPM



- Package and printed PCB designers needs :
  - Accurate and relatively simple IC power model
  - Impedance and resonant frequencies of the global PDN
  - Accurate current waveforms to represent a realistic dynamic scenario for the chip.

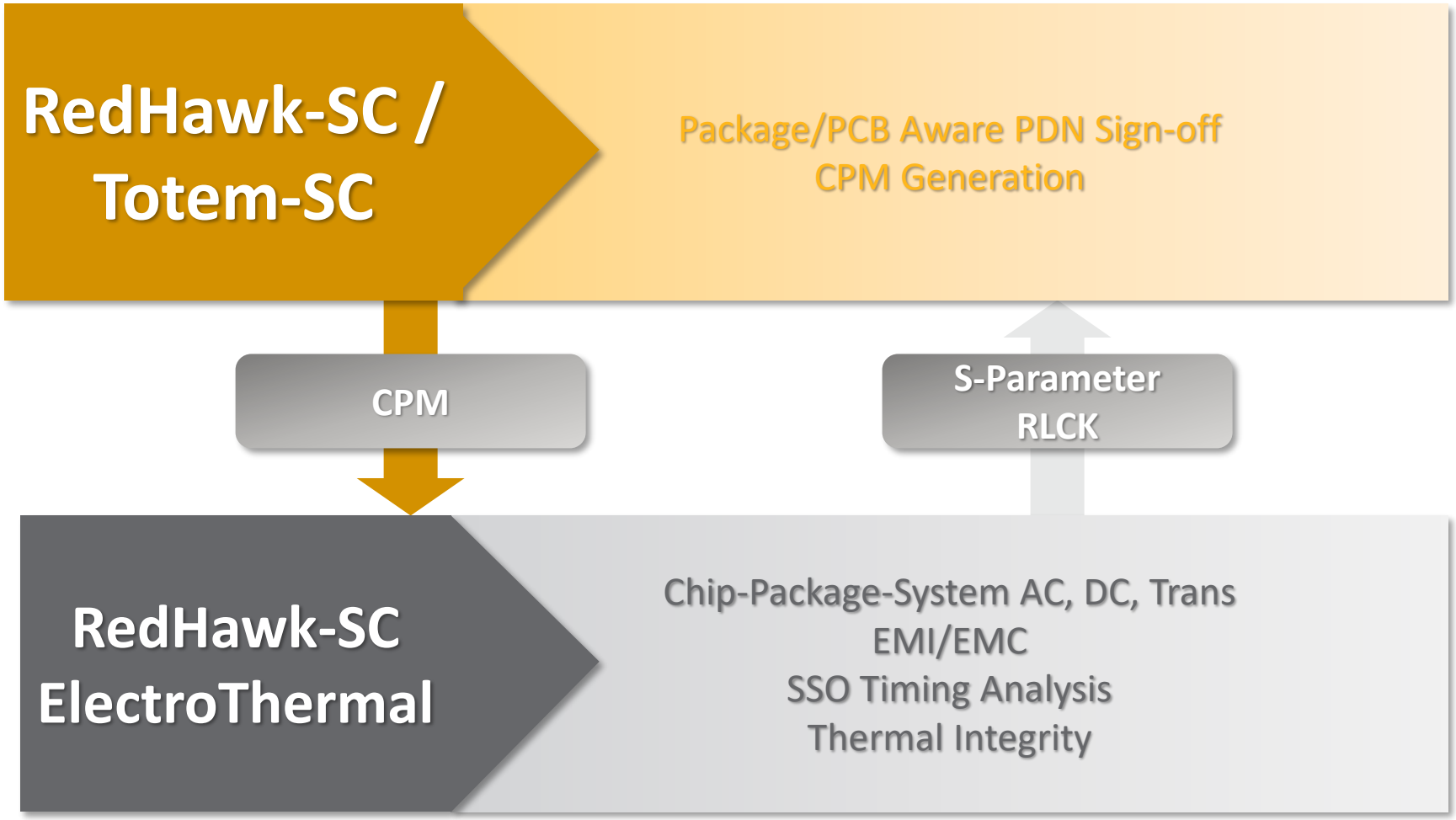
- Ansys RedHawk-SC CPM technology can meet above requirements of Package and PCB designers

CPM Cdie vs Ctotal



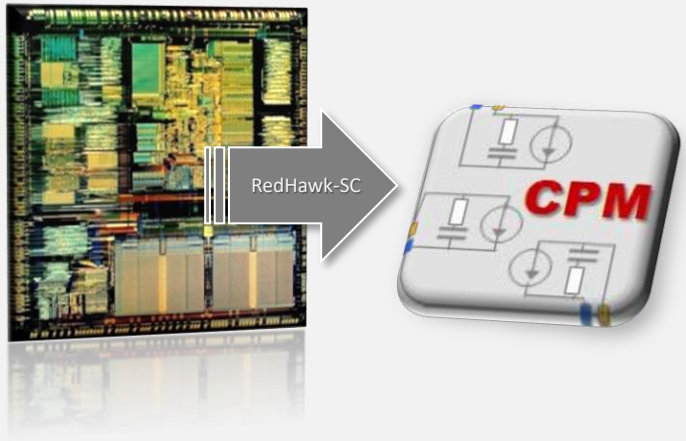
C_{die}	C_{total}
Effective capacitance looking into the die	Algebraic summation of all the on-die capacitances
Can be measured	Can't be measured
Frequency dependent , close to C_{total} at lower frequency range	Not Frequency dependent

Ansys CPS Solutions



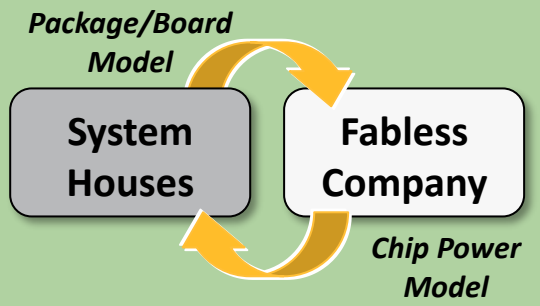
What’s in a CPM ?

What's in a CPM?



- ✓ Chip on-die Power Grid RLC
- ✓ Transistor current/cap/ESR
- ✓ Open SPICE netlist format

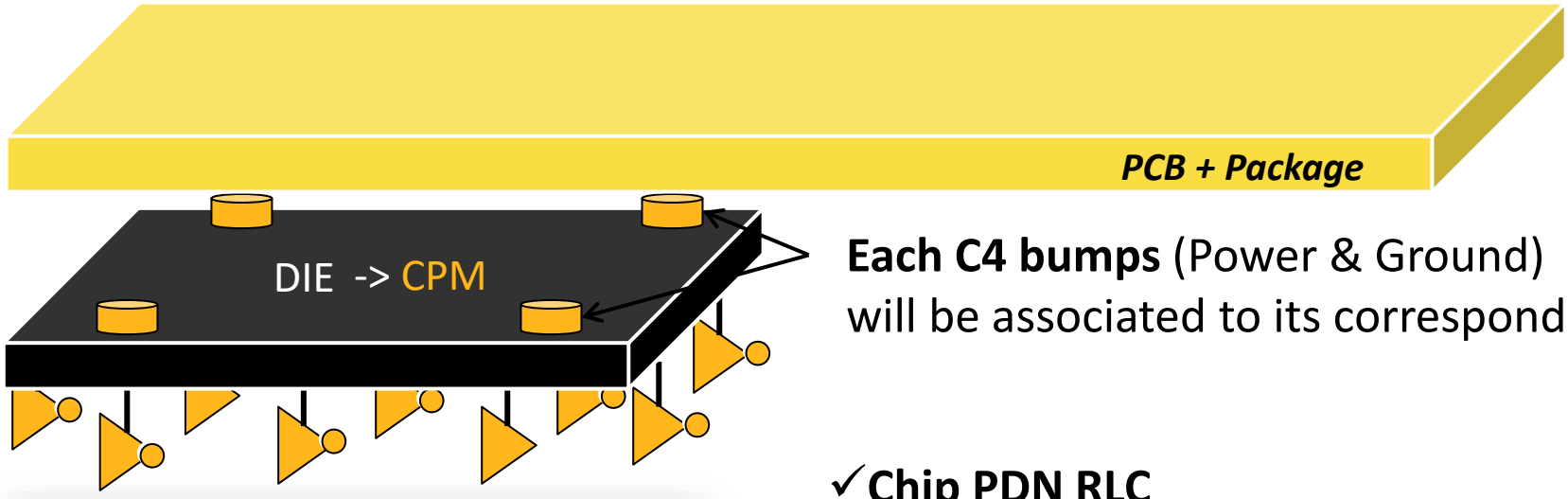
- ➢ **Multi-domain, distributed model**
- ➢ **DC to multi-GHz validity**
- ➢ **Advanced chip excitation modes**
- ➢ **Silicon correlated**



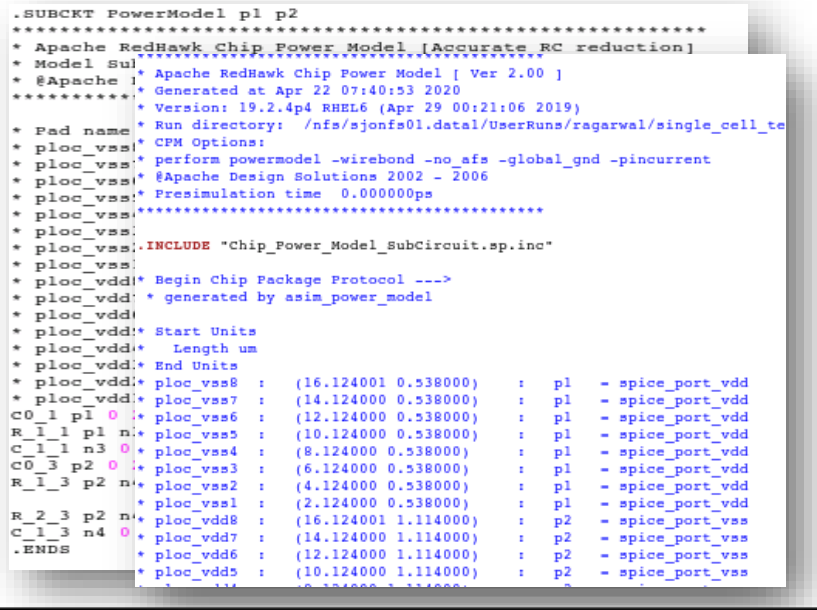
What's in a CPM?

- **Full chip frequency domain simulation and model order reduction**
- **Cdie/Rdie for each domain**
- **Icc(t) for every domain and pin**
  - Full chip current signature simulation
  - VCD based and Vectorless switching scenario

What's in a CPM?



Each C4 bumps (Power & Ground) will be associated to its corresponding



```
.SUBCKT PowerModel p1 p2
* Apache RedHawk Chip Power Model [Accurate RC reduction]
* Model Su
* @Apache
* Pad name
* ploc_vss
* ploc_vss
* ploc_vss
* ploc_vss
* ploc_vss
* ploc_vss
* ploc_vss
* ploc_vss
* ploc_vdd
* ploc_vdd
* ploc_vdd
* ploc_vdd
* ploc_vdd
* ploc_vdd
* ploc_vdd
* ploc_vdd
C0_1 p1 0
R_1_1 p1 n
C_1_1 n3 0
C0_3 p2 0
R_1_3 p2 n
R_2_3 p2 n
C_1_3 n4 0
.ENDS
```

```
* Apache RedHawk Chip Power Model [ Ver 2.00 ]
* Generated at Apr 22 07:40:53 2020
* Version: 19.2.4p4 RHEL6 (Apr 29 00:21:06 2019)
* Run directory:  /nfs/sjonfs01.datal/UserRuns/ragarwal/single_cell_te
* CPM Options:
* perform powermodel -wirebond -no_afs -global_gnd -pincurrent
* @Apache Design Solutions 2002 - 2006
* Presimulation time  0.000000ps
*****************************************

.INCLUDE "Chip_Power_Model_SubCircuit.sp.inc"

* Begin Chip Package Protocol --->
* generated by asim_power_model

* Start Units
*   Length um
* End Units
* ploc_vss8  :   (16.124001 0.538000)    :   p1   = spice_port_vdd
* ploc_vss7  :   (14.124000 0.538000)    :   p1   = spice_port_vdd
* ploc_vss6  :   (12.124000 0.538000)    :   p1   = spice_port_vdd
* ploc_vss5  :   (10.124000 0.538000)    :   p1   = spice_port_vdd
* ploc_vss4  :   (8.124000 0.538000)     :   p1   = spice_port_vdd
* ploc_vss3  :   (6.124000 0.538000)     :   p1   = spice_port_vdd
* ploc_vss2  :   (4.124000 0.538000)     :   p1   = spice_port_vdd
* ploc_vss1  :   (2.124000 0.538000)     :   p1   = spice_port_vdd
* ploc_vdd8  :   (16.124001 1.114000)    :   p2   = spice_port_vss
* ploc_vdd7  :   (14.124000 1.114000)    :   p2   = spice_port_vss
* ploc_vdd6  :   (12.124000 1.114000)    :   p2   = spice_port_vss
* ploc_vdd5  :   (10.124000 1.114000)    :   p2   = spice_port_vss
```

✓**Chip PDN RLC**

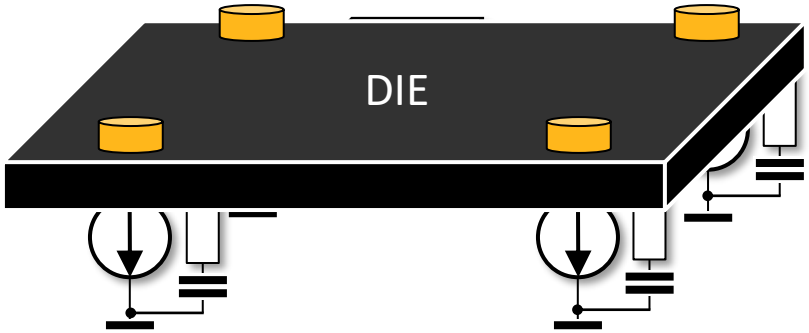
Physical model of chip layout

✓**Transistor/cell current /cap/ESR**

Electrical model of chip layout

✓ **CPM is topological, Physical and activity based**

What's in a CPM?



Per port :
current source
RLC
coupling

- ✓ Each port (or bump) reflects the current flow associated with that port (or bump) reflecting the on-die activity
- ✓ Parasitics are associated with every port (or bump)
- ✓ Each port (or bump) are coupled with every other port

```
.SUBCKT PowerModel p1 p2
*************************************************************
* Apache RedHawk Chip Power Model [Accurate RC reduction]
* Model Subcircuit of Die PDN
* @Apache Design Solutions 2007
*************************************************************

* Pad name : port name : net : port id
* ploc_vss8    spice_port_vdd    vdd       p1
* ploc_vss7    spice_port_vdd    vdd       p1
* ploc_vss6    spice_port_vdd    vdd       p1
* ploc_vss5    spice_port_vdd    vdd       p1
* ploc_vss4    spice_port_vdd    vdd       p1
* ploc_vss3    spice_port_vdd    vdd       p1
* ploc_vss2    spice_port_vdd    vdd       p1
* ploc_vss1    spice_port_vdd    vdd       p1
* ploc_vdd8    spice_port_vss    vss       p2
* ploc_vdd7    spice_port_vss    vss       p2
* ploc_vdd6    spice_port_vss    vss       p2
* ploc_vdd5    spice_port_vss    vss       p2
* ploc_vdd4    spice_port_vss    vss       p2
* ploc_vdd3    spice_port_vss    vss       p2
* ploc_vdd2    spice_port_vss    vss       p2
* ploc_vdd1    spice_port_vss    vss       p2
c0_1 p1 0 2.9222805485e-15
R_1_1 p1 n3 7.78743720829
c_1_1 n3 0 3.97347104531e-14
c0_3 p2 0 2.9222805485e-15
R_1_3 p2 n4 7.78747807385

R_2_3 p2 n4 1484000
c_1_3 n4 0 3.97347104531e-14
.ENDS
```

spice netlist

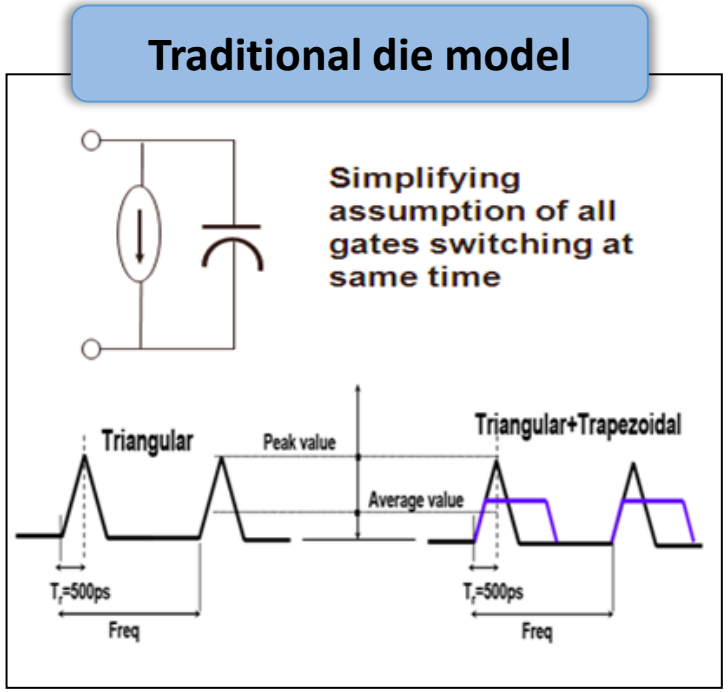
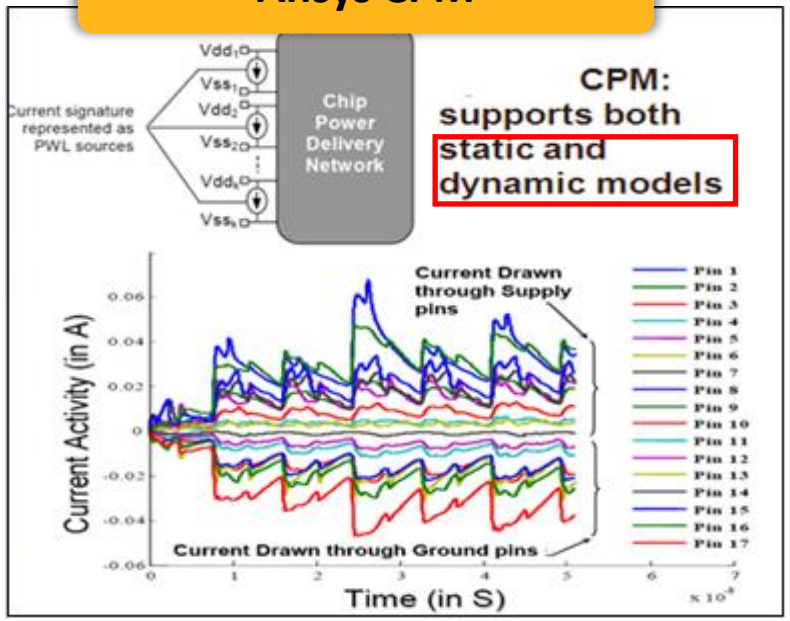
Passive RC Values

```
*****************************************
* Apache RedHawk Chip Power Model [ Ver 2.00 ]
* Generated at Apr 22 07:40:53 2020
* Version: 19.2.4p4 RHEL6 (Apr 29 00:21:06 2019)
* Run directory:  /nfs/sjonfs01.datal/UserRuns/ragarwal/si...le_cell_te
* CPM Options:
* perform powermodel -wirebond -no_afs -global_gnd -pincurrent
* @Apache Design Solutions 2002 - 2006
* Presimulation time  0.000000ps
*****************************************

.INCLUDE "Chip_Power_Model_SubCircuit.sp.inc"

* Begin Chip Package Protocol --->
 * generated by asim_power_model

* Start Units
*   Length um
* End Units
* ploc_vss8  :   (16.124001 0.538000)    :   p1   = spice_port_vdd
* ploc_vss7  :   (14.124000 0.538000)    :   p1   = spice_port_vdd
* ploc_vss6  :   (12.124000 0.538000)    :   p1   = spice_port_vdd
* ploc_vss5  :   (10.124000 0.538000)    :   p1   = spice_port_vdd
* ploc_vss4  :   (8.124000 0.538000)     :   p1   = spice_port_vdd
* ploc_vss3  :   (6.124000 0.538000)     :   p1   = spice_port_vdd
* ploc_vss2  :   (4.124000 0.538000)     :   p1   = spice_port_vdd
* ploc_vss1  :   (2.124000 0.538000)     :   p1   = spice_port_vdd
* ploc_vdd8  :   (16.124001 1.114000)    :   p2   = spice_port_vss
* ploc_vdd7  :   (14.124000 1.114000)    :   p2   = spice_port_vss
* ploc_vdd6  :   (12.124000 1.114000)    :   p2   = spice_port_vss
* ploc_vdd5  :   (10.124000 1.114000)    :   p2   = spice_port_vss
```

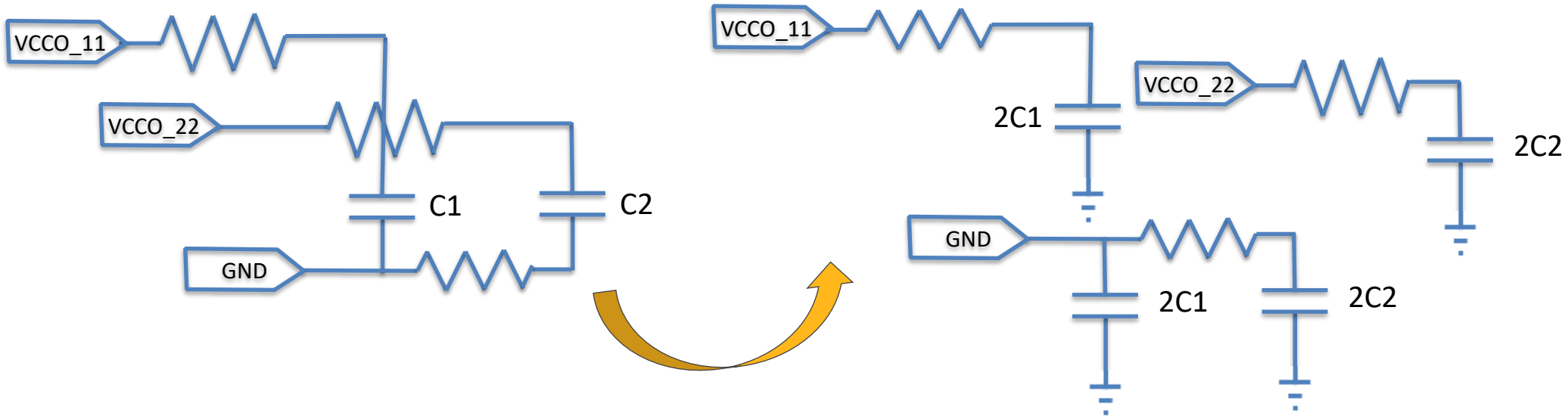
Active Current Signature

average current ?

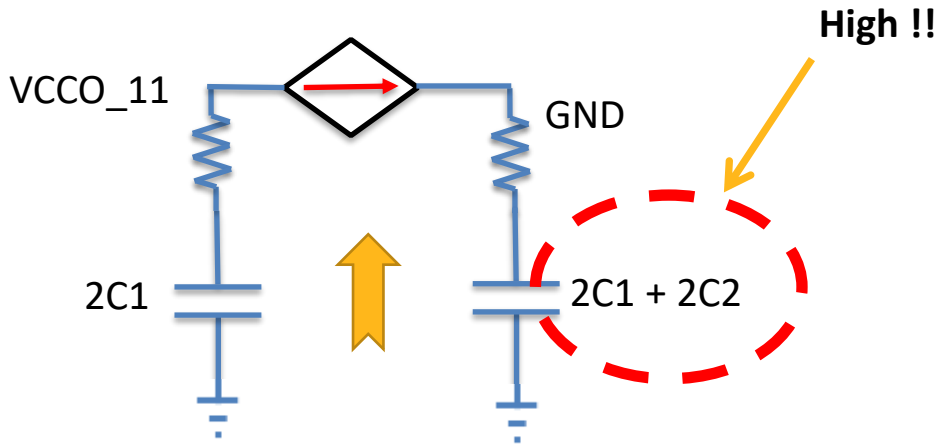
Benefits of CPM

	Traditional die model	Ansys CPM
Chip Current		
Chip Parasitics	Estimated Cdie Missing Rdie, Ldie **Single Lumped Model**	Power-grid RLC Intrinsic,Intentional De-cap, Load cap Spatial and Temporal switching current **CPM includes all die parasitics**

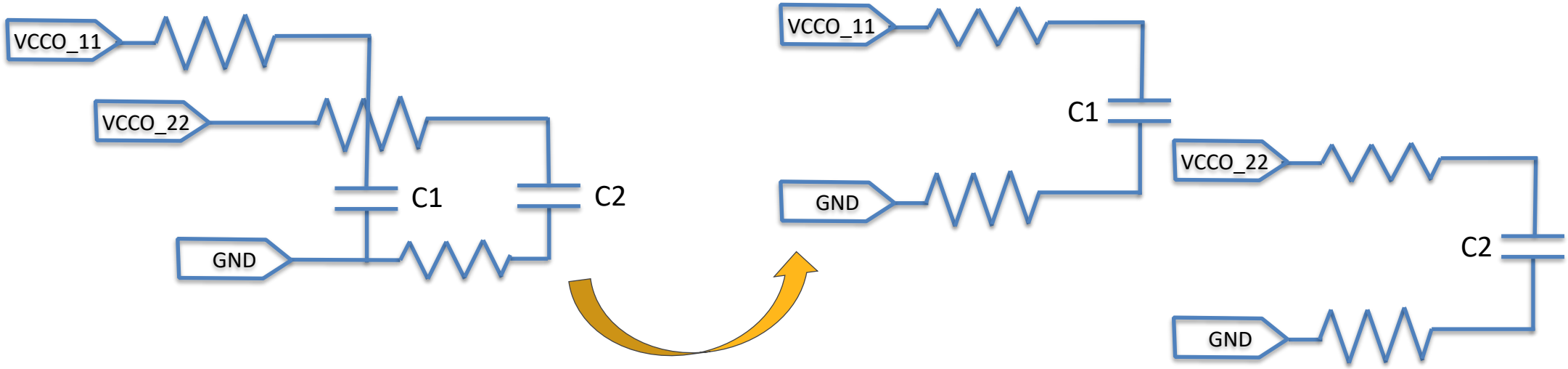
De-coupled Solver (Default)



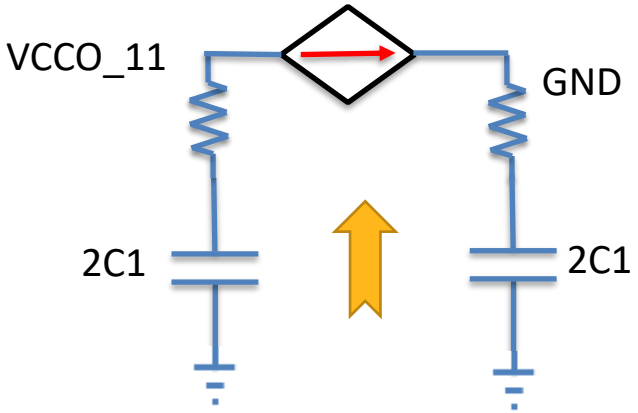
- By default, CPM solver will consider decoupled Caps
- Hence GND will include caps corresponding to other domains as well.
- This approximation is good for single PG domain chips with symmetric package



Coupled Solver

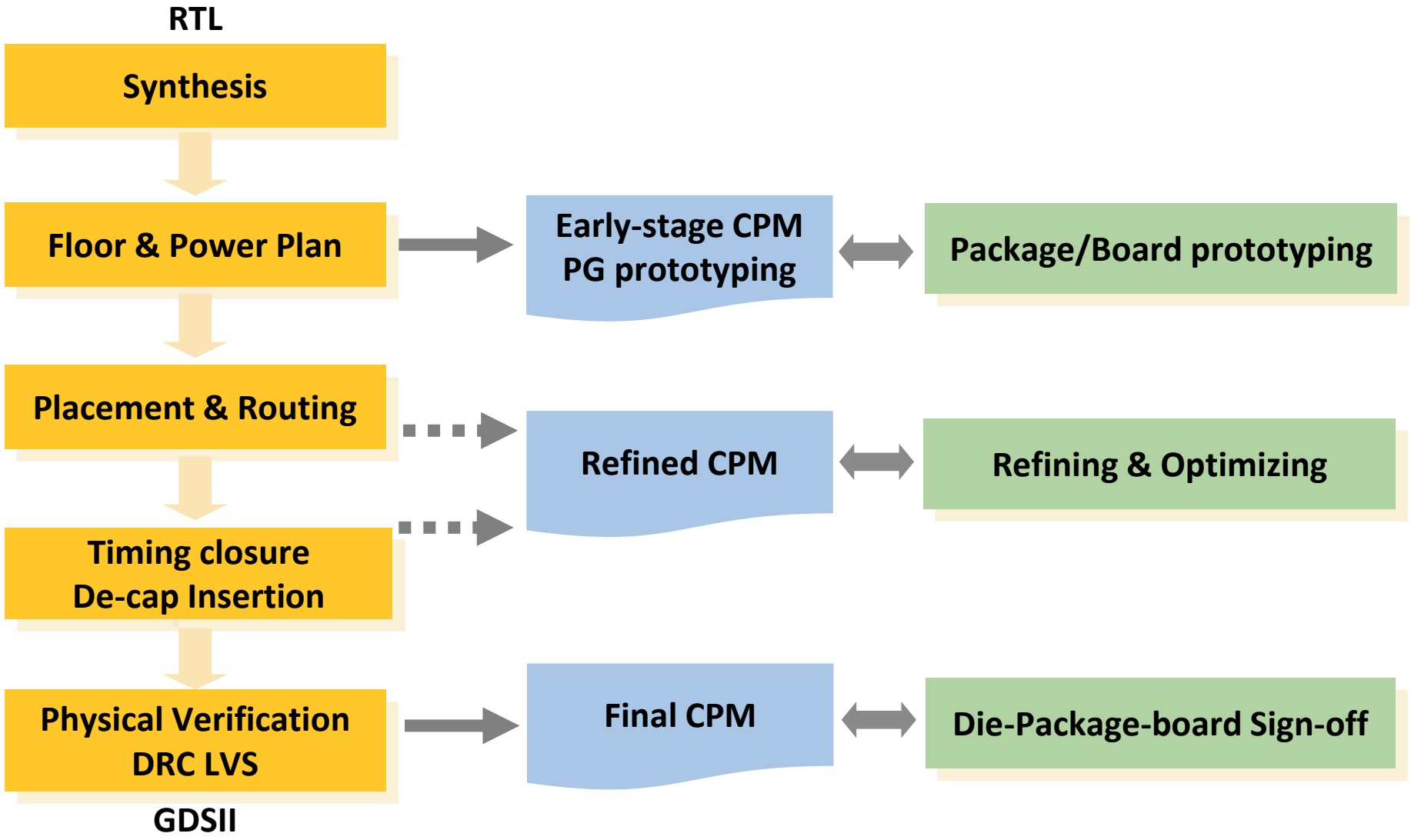


- CPM solver will consider coupled Caps between respective domains.
- Good for multiple PG domain and asymmetric package
- Coupled solver is expected to take more runtime



Chip Power Model (CPM) Use

CPM based IC-System Co-Design

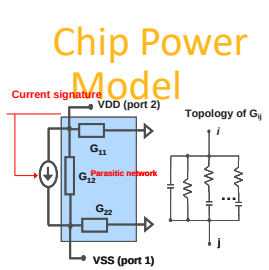


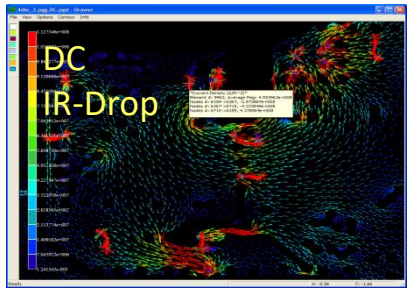
IC-Aware Package/PCB Power Integrity Solution

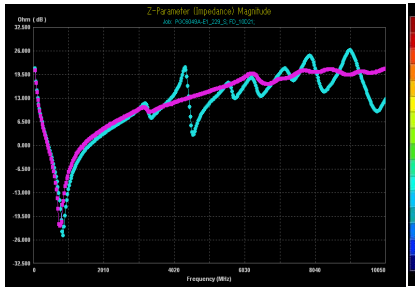
CPM as used by pkge designers

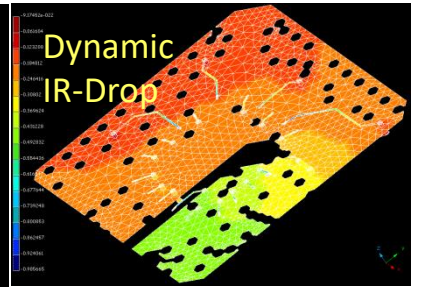
IC-Aware Package-PCB power integrity

- Native import of CPM and automated connection to package
- Static IR drop (DC) analysis: CPM can be used as current sink
- Frequency-domain AC simulation: Review CPM+Package+PCB Impedance response
- Dynamic analysis (Transient) : Using time-varying current for each die port from CPM



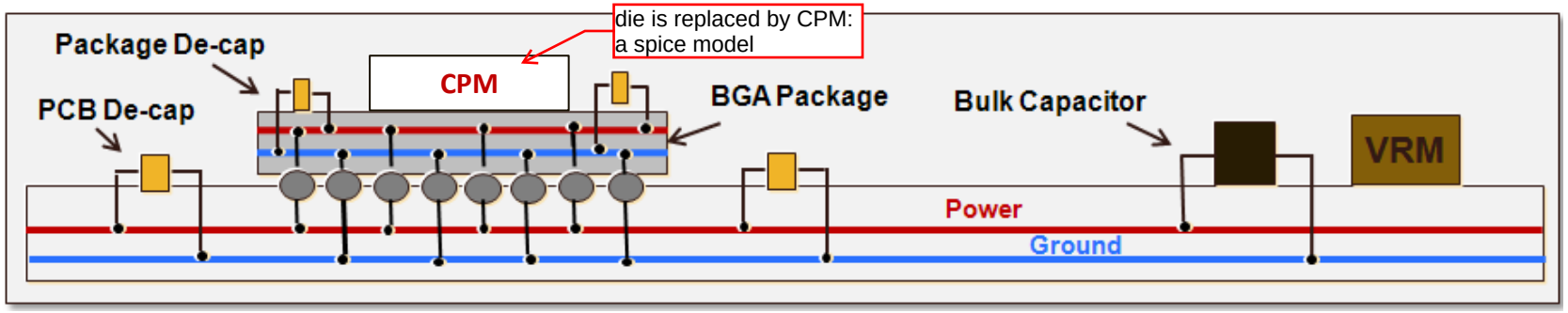


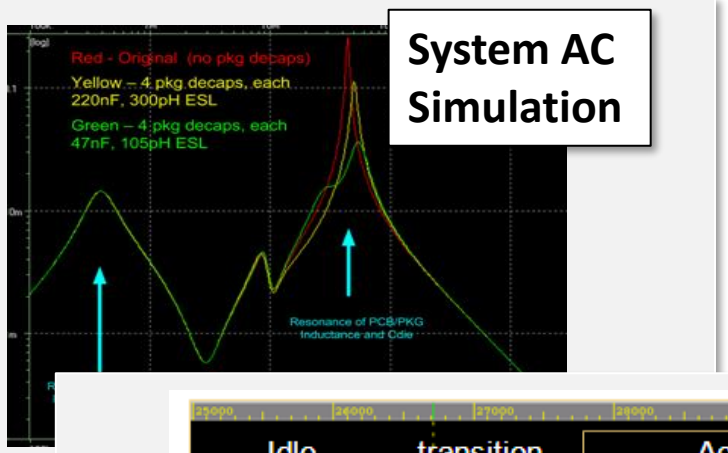




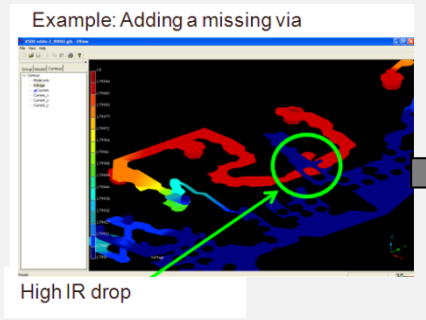
Customized analysis in a single GUI environment

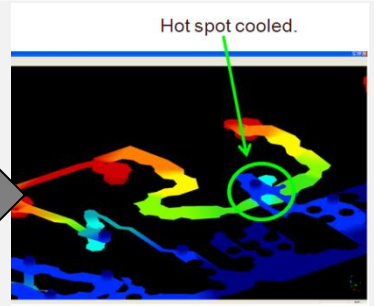
Enabling Global PDN Analysis



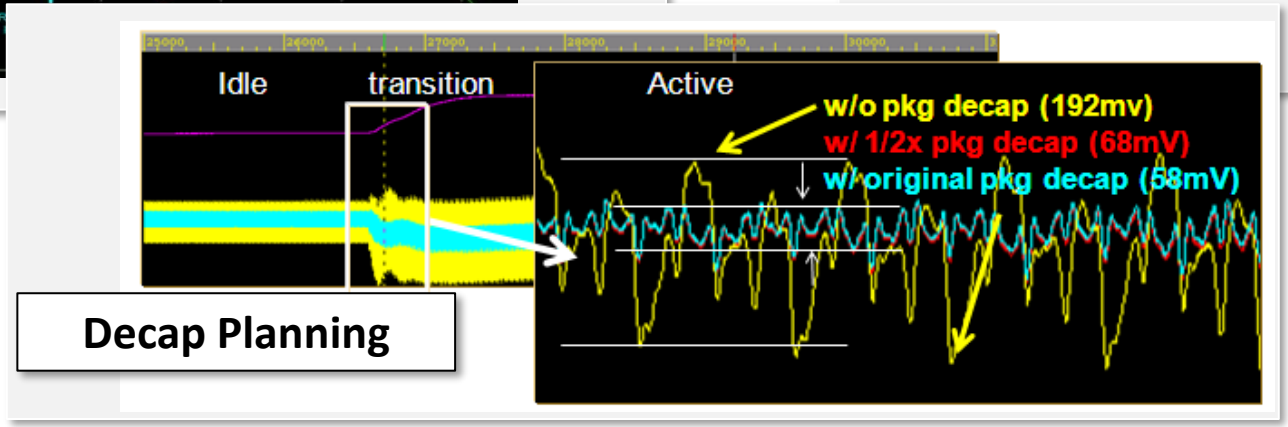


System AC Simulation





Package/PCB DC Analysis

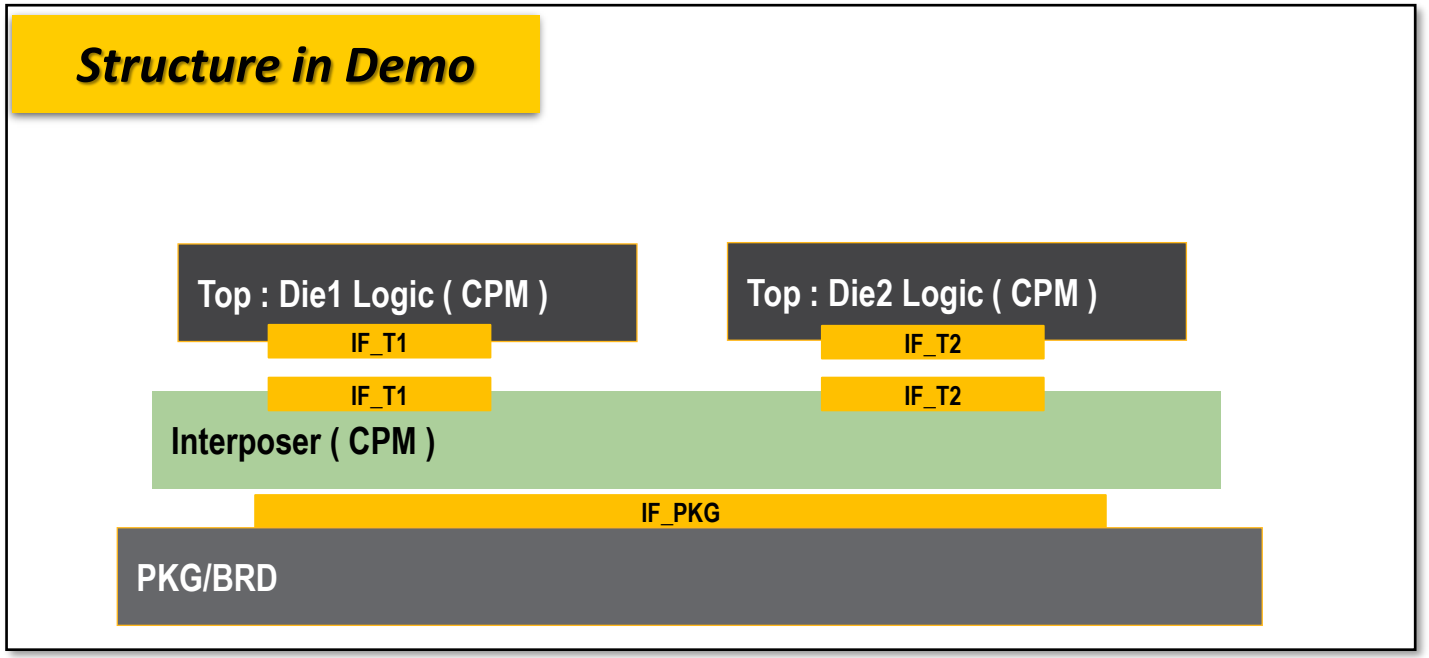


Decap Planning

Transient System Simulation

Enabling 3D-IC Power Integrity Analysis

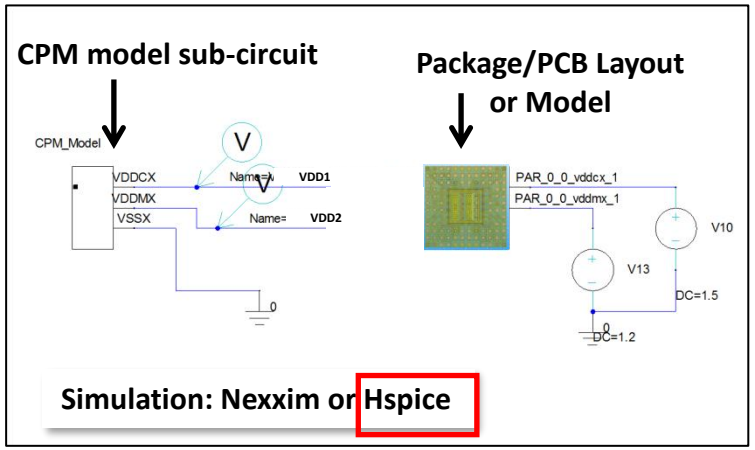
- One or More Die Converted to CPM
- Reduces runtime by abstracting the design
- For IP reasons, Die can come from 3rd Party
- CPM has only electrical current and PDN representation

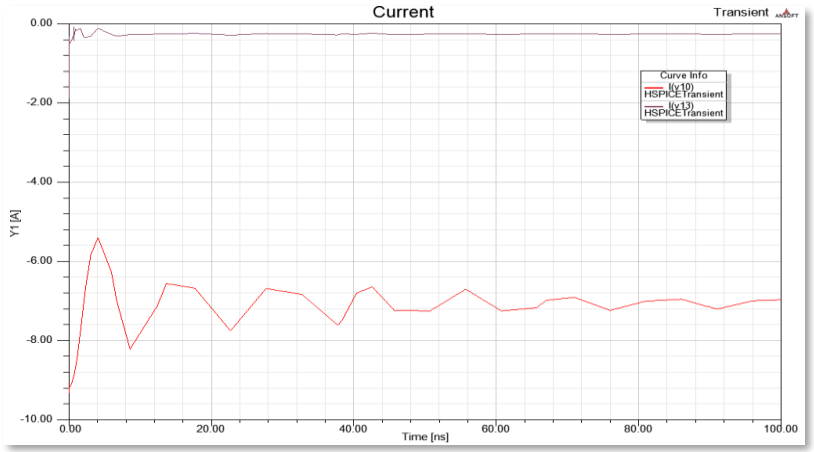


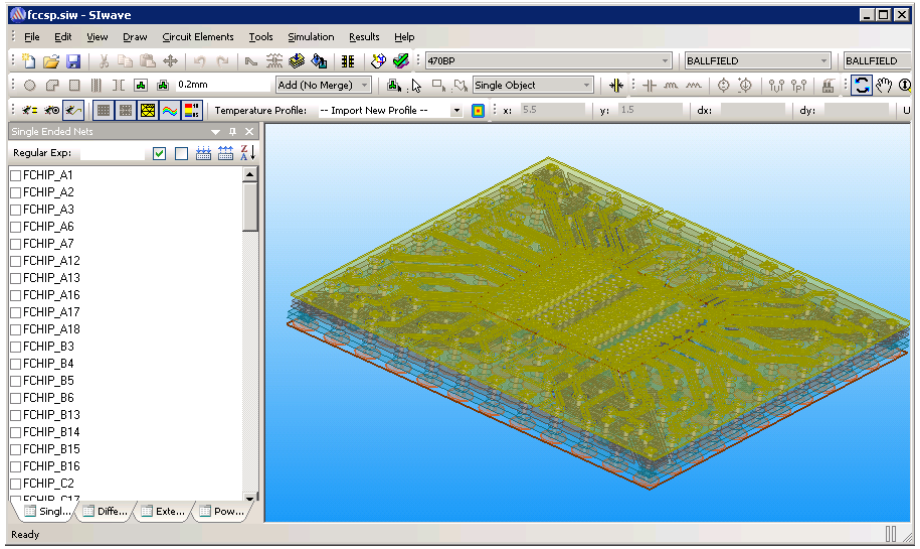
Design Structure :

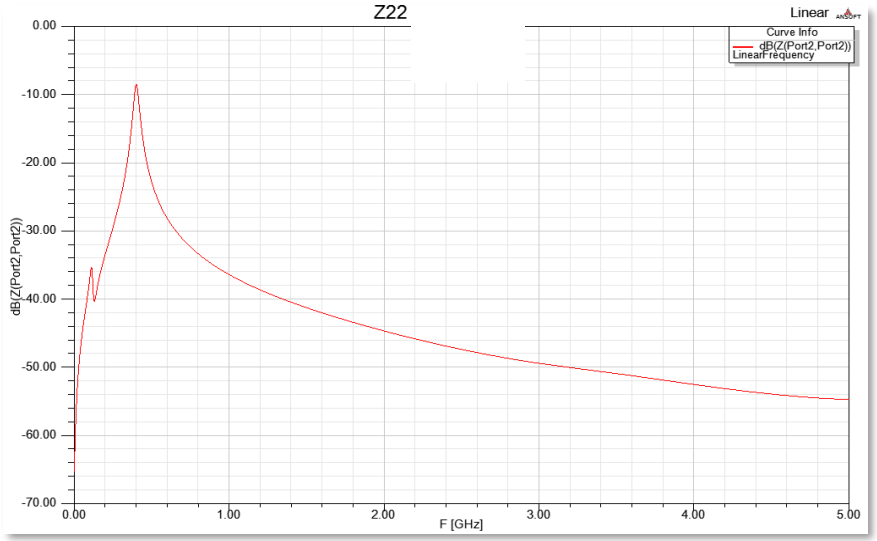
Logic Processor (Chip Power Model)
Interposer (Chip Power Model) +
PKG + Board (Extracted Model)

Using CPM in Ansoft Designer









Getting Familiar with Labs

Getting Familiar with Modular Training Labs

Lab Instructions :

- Download the Galaxy_Training.tar.gz Training Bundle
- Move to Modular_Training/07_CPM_Creation directory

Getting Familiar with Modular Training Labs

Lab Scripts Organization :

Top Wrapper Script

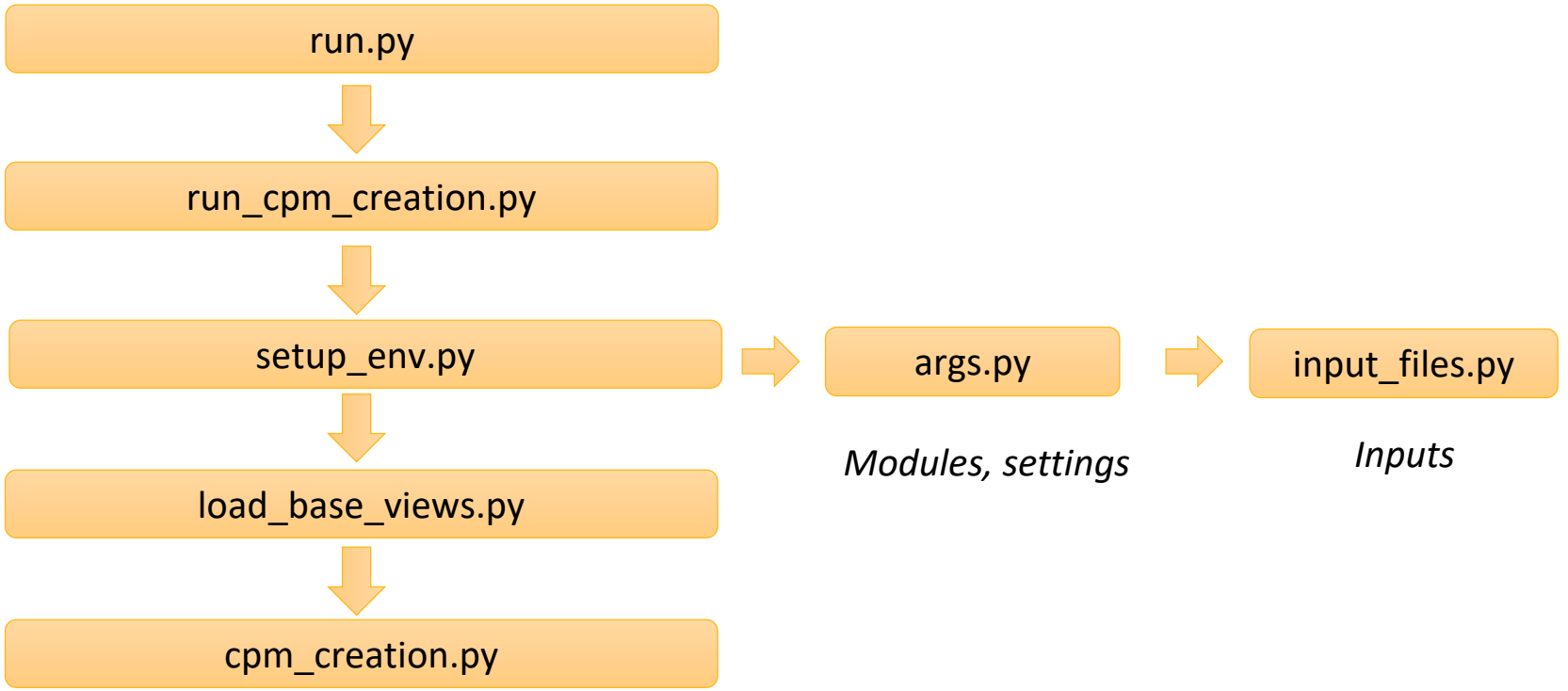
Script that runs all steps

Has modules , settings , input files

loads base views – dv, ev, evx, tv

Create Views like :

- *Rollup ExtractView*
- *Grm type SimulationView*
- *GrmView*



Getting Familiar with Modular Training Lab Scripts

File : run.py

```
include('../scripts/run_cpm_creation.py')
```

File : ../scripts/run_cpm_creation.py

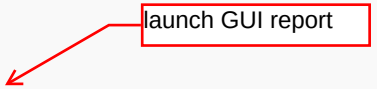
```
include('setup_env.py')
include('load_base_views.py')
include('cpm_creation.py')
```

- setup_env.py loads worker environment, variables & arguments
- load_base_views.py script loads the base views like dv, ev
- cpm_creation.py creates all the required views and reports

Setting up the environment

File : ../scripts/setup_env.py

```
import pprint
open_scheduler_window()

design_data_path = '../../design_data/'

ll = create_local_launcher('local')
register_default_launcher(ll, min_num_workers=10)

include('args.py')
```



Set *design_data_path* variable to central design data path area

Create Launcher for Workers

Set the arguments for various view creation commands in args.py

Launching RedHawk-SC - Get started with your Modular Training

- **Batch mode execution example:**
  - `<path_to_rhsc_installation>/bin/redhawk_sc run.py`
- **Interactive mode execution example:**
  - `<path_to_rhsc_installation>/bin/redhawk_sc -i run.py`
  - It needs an exit() command in script or entered manually to exit the Python shell
- **Connecting to a live RedHawk-SC run:**
  - RedHawk-SC allows querying of data/results from an active session, by remotely attaching to the session
  - Multiple users can attach to the same session from multiple machines for querying/viewing results
  - `<path_to_rhsc_installation>/bin/redhawk_sc -r <gp_dir>`
- **Execution Log Files:**
  - All RHSC log files reside by default in gp<> folder
    - If the run is fired in the same directory, tool will create gp.1, gp.2 incrementally
    - 'latest.gp' link will point to the most recent gp directory
  - Main log file for RedHawk-SC would be <gp_directory>/run.log file.

Chip Power Model (CPM) Generation

Ansys

Generating CPM

Extract View Rollup

- Rolls up the design to higher metal layer
- For more details , see :

  help(SeaScapeDB.create_extract_view)

SimulationView

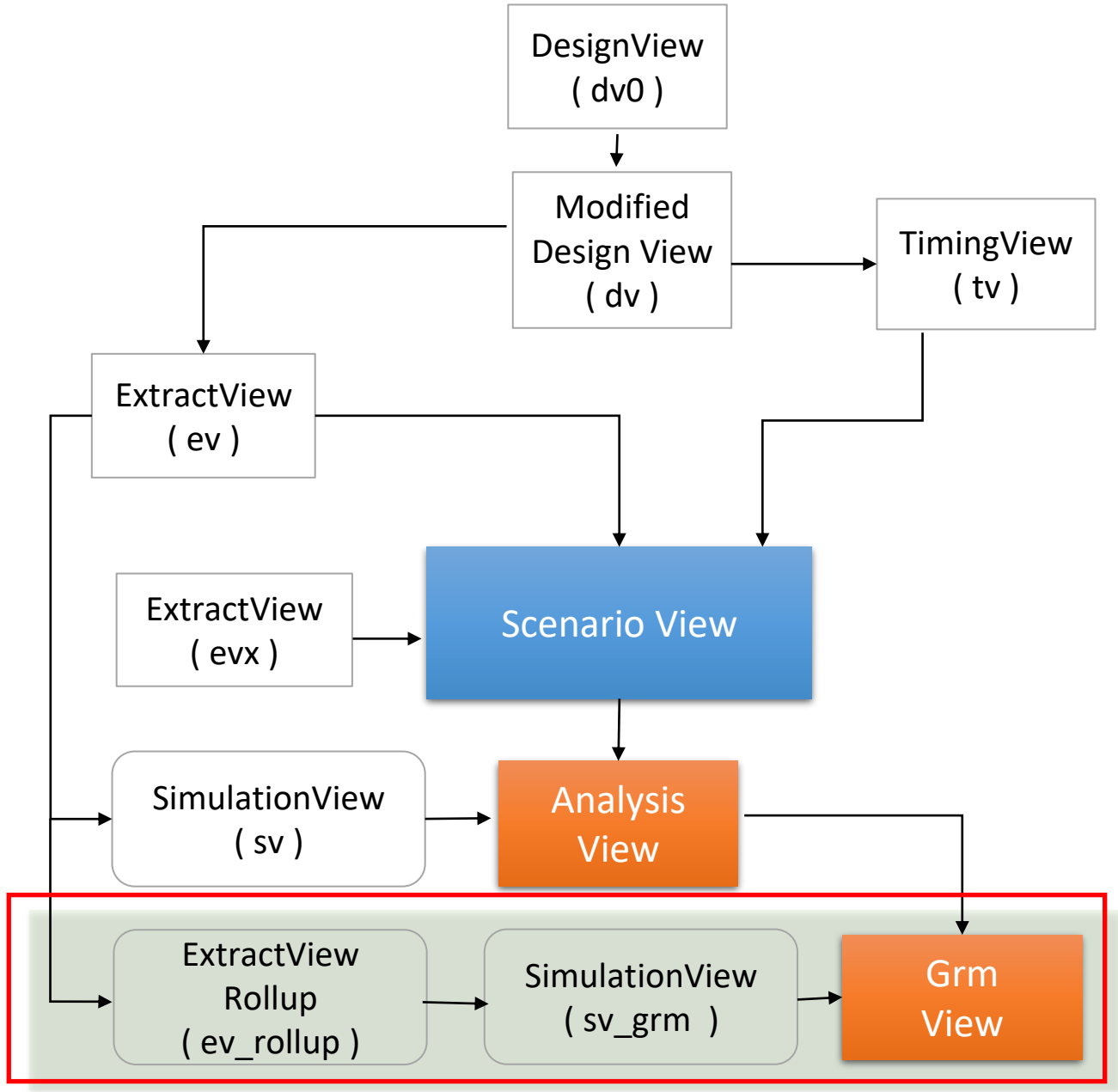
- Sim_type CPM is required
- For more details , see :

  help(SeaScapeDB.create_simulation_view)

GrmView

- GrmView (generic reduced model)
- Takes in SimulationView and AnalysisView as inputs
- Generates CPM Model files
- For more details , see :

  help(SeaScapeDB.create_reduced_model_view)



Big Worker Setup

- Currently CPM creating engine is a single worker job
- It may consume high memory in big designs
- It is advised to launch a big memory worker and allocate it for cpm jobs
- Required only for large design

```
big_launcher = create_uge_launcher('big_launcher' ,'qsub -V -b y -cwd-j y -q ae_perf -l
mfree=250G -o uge.log2')
big_launcher.set_jobs(['cpm.write_spice_deck*', 'cpm.run_asim_power_model*'])
big_launcher.launch(1)
```

setup worker settings

direct tasks to worker

Note : This is an example when UGE is the clustering software. LSF would require a slightly different syntax.

Using Rollup for CPM

- Currently CPM creating engine is a single worker job
- Rollup techniques may be required in large designs
  - It reduces the node count and more importantly the memory usage.
  - No loss of accuracy.
- Metal Rollup is already available in ev , we need to pass rollup_settings to ExtractView :

```
import ev_utils
ev_rollup =db.create_extract_view( ... ,
rollup_settings=ev_utils.create_rollup_settings(dv,keep_metals=2),...)
```

- Here keep_metals = 2 will keep top 2 layers in design and rollup all other layers
- For MiMCap Design , keep_metals = 5 can be used to retain MiM Cap structure

*For details on rollup technique kindly refer to Rollup Training

Preparing for CPM run

File : scripts/args.py

```
ev_rollup_args = dict(
   temperature=25,
   tag='ev_rollup',
   options=options)

sv_grm_args = dict(
   options=options,
   sim_type='cpm',
   tag='sv_grm')

gv_args = dict(
   options=options,
   tag='gv',
   cpm_nx = 2 ,
   cpm_ny = 2 ,     )
```

divide chip to 2 partition in x dir,
divide chip to 2 partition in y dir

In one partition:
all pwr plocs mapped to 1 pwr port,
all gnd plocs mapped to 1 gnd port

File : scripts/args.py

```
gv_coupled_args = dict(
   options=options,
   tag='gv_coupled',
   coupled_simulation = True,
   cpm_nx = 2 ,
   cpm_ny = 2 ,
   )
```

Turn on coupled CPM generation

For 2x2 CPM

Preparing for CPM run

File : scripts/cpm.py

this file is missing from lab ??

sv: simulation view
gv: grm view
ev: extract view
dv
av

these are defined in args.py

intermediate var.

Top-2 rollup is used here

For Decoupled CPM

For Coupled CPM

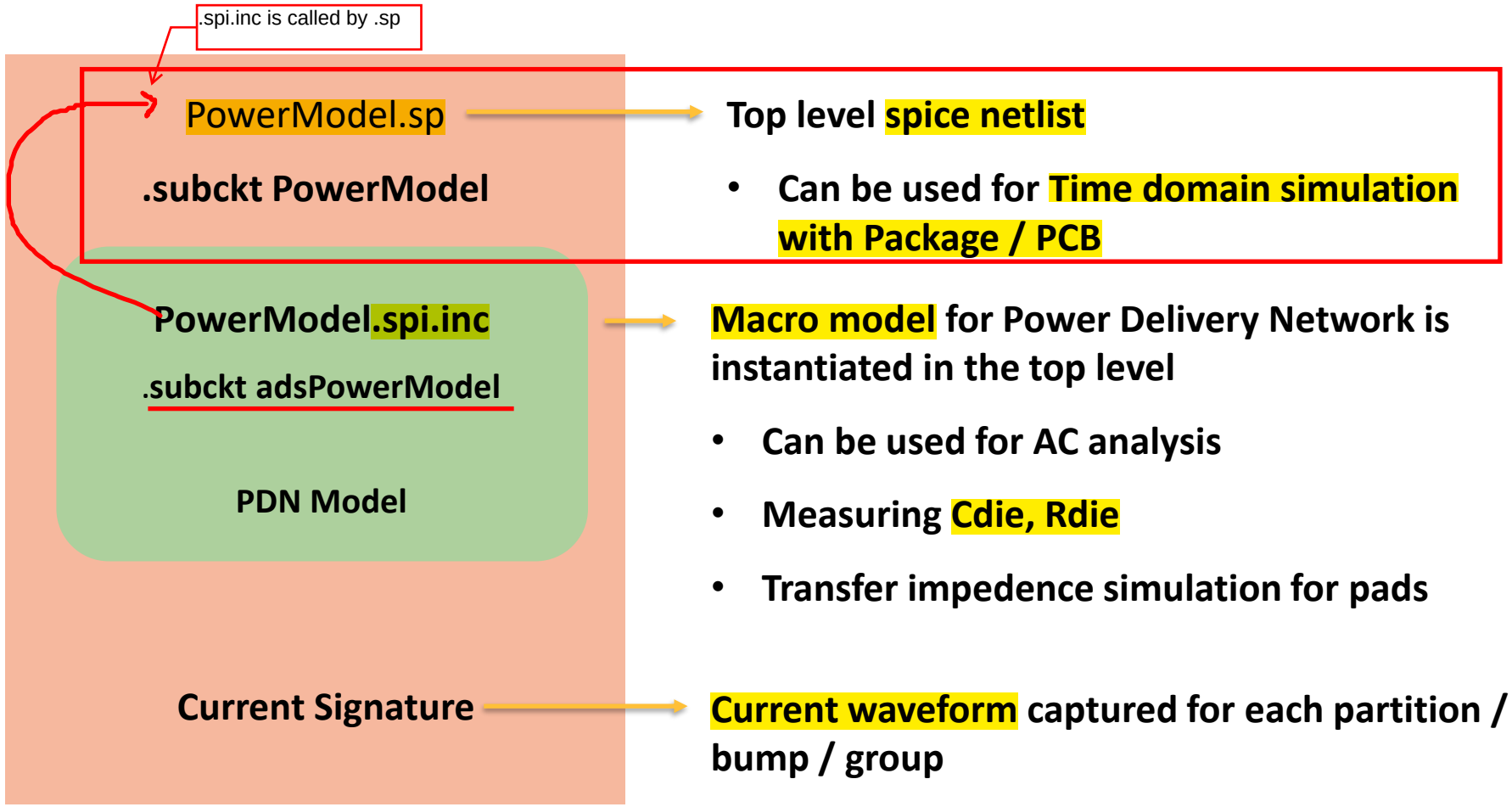
work horse function

output file

```
sv_wo_pkg  = db.create_simulation_view(ev, package=None, **sv_wo_pkg_args)
av_dynamic_wo_pkg = db.create_analysis_view(sv_wo_pkg,
scn_no_prop_vectorless, **av_dynamic_wo_pkg_args)
ev_rollup = db.create_extract_view(design_view =
dv,rollup_settings=ev_utils.create_rollup_settings(dv,keep_metals=2),tech_v
iew=nv,**ev_rollup_args)
sv_grm     = db.create_simulation_view(extract_view=ev_rollup,**sv_grm_args)
gv         =
db.create_reduced_model_view(simulation_view=sv_grm,analysis_view =
av_dynamic_wo_pkg,**gv_args)
gv_coupled         =
db.create_reduced_model_view(simulation_view=sv_grm,analysis_view =
av_dynamic_wo_pkg,**gv_coupled_args)

current_spice, passive_spice = gv.get_reduced_model_in_cpm_spice()
write_to_file('Chip_Power_Model.sp', current_spice)
write_to_file('Chip_Power_Model_SubCircuit.sp.inc', passive_spice)
write_to_file('perform_ac.sp',gv.get_reduced_model_perform_ac_spice())
```

CPM Outputs

- Hierarchical Structure of CPM Output File



Command to get the CPM Output files

- GrmView.get_reduced_model_in_cpm_spice()
  - Returns Current Spice and Passive Spice file

```
current_spice, passive_spice = gv.get_reduced_model_in_cpm_spice()

write_to_file('Chip_Power_Model.sp', current_spice)

write_to_file('Chip_Power_Model_SubCircuit.sp.inc', passive_spice)
```

8 ports

```
.SUBCKT PowerModel p1 p2 p3 p4 p5
+ p6 p7 p8
**********************************************************
* Ansys RedHawk-SC Chip Power Model [Accurate RC reduction]
* Model Subcircuit of Die PDN
* Copyright (c) 2002-2020 ANSYS, Inc.
**********************************************************

* Pad name : port name : net : port id
* VDD_56   VDD_56   VDD      p1
* VDD_160   VDD_160   VDD      p1
* VDD_38   VDD_38   VDD      p1
* VDD_159   VDD_159   VDD      p1
* VDD_158   VDD_158   VDD      p1
* VDD_157   VDD_157   VDD      p1
* VDD_4   VDD_4   VDD      p1
* VDD_5   VDD_5   VDD      p1
```

Die Power Delivery Network (PDN) Model Subckt

diff port_names can share same port_id

these plocs mapped to the same p1 port

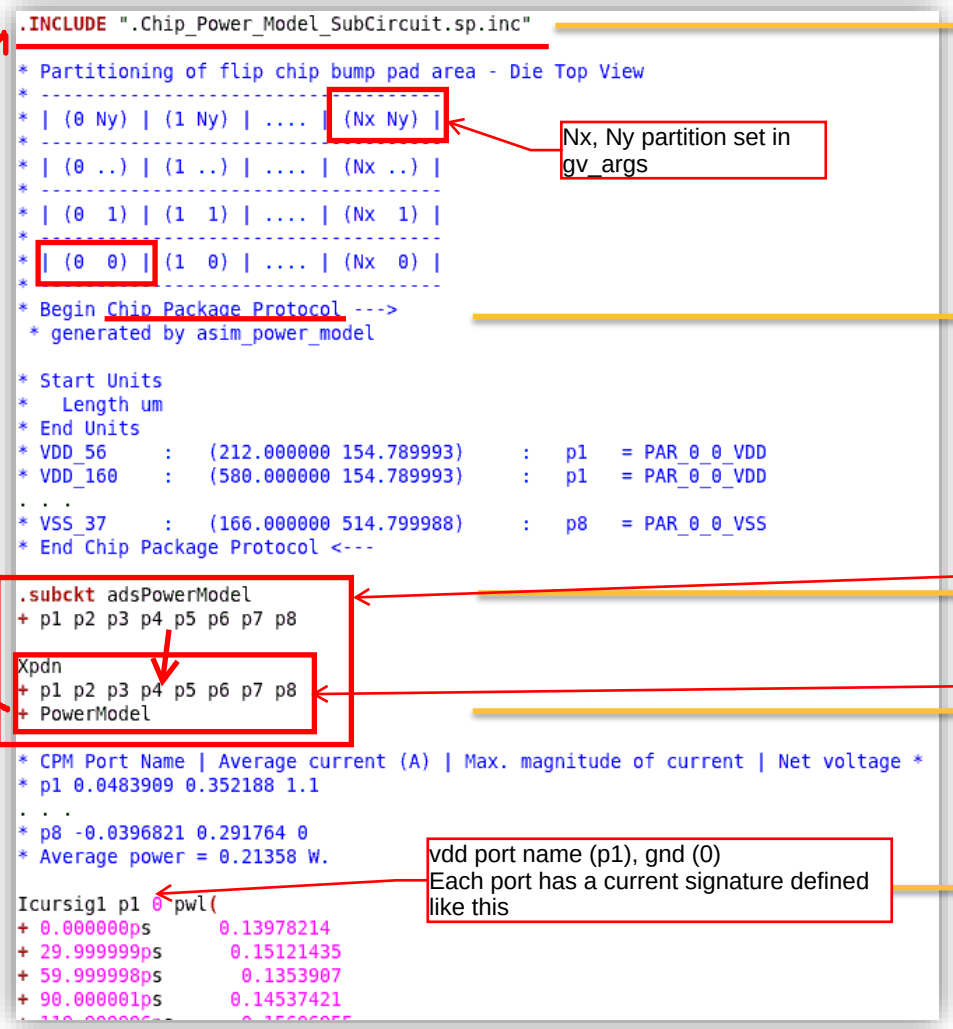
after this part, the reduced PDN RLC model in spice format

Passive Spice : Chip_Power_Model_SubCircuit.sp.inc

.ENDS

Command to get the CPM Output files

.sp SPICE circuit



```
.INCLUDE ".Chip_Power_Model_SubCircuit.sp.inc"

* Partitioning of flip chip bump pad area - Die Top View
* ------------------------------------
* | (0 Ny) | (1 Ny) | .... | (Nx Ny) |
* ------------------------------------
* | (0 ..) | (1 ..) | .... | (Nx ..) |
* ------------------------------------
* | (0  1) | (1  1) | .... | (Nx  1) |
* ------------------------------------
* | (0  0) | (1  0) | .... | (Nx  0) |
* ------------------------------------
* Begin Chip Package Protocol --->
 * generated by asim_power_model

* Start Units
*   Length um
* End Units
* VDD_56     :   (212.000000 154.789993)     :   p1   = PAR_0_0_VDD
* VDD_160    :   (580.000000 154.789993)     :   p1   = PAR_0_0_VDD
. . .
* VSS_37     :   (166.000000 514.799988)     :   p8   = PAR_0_0_VSS
* End Chip Package Protocol <---

.subckt adsPowerModel
+ p1 p2 p3 p4 p5 p6 p7 p8

Xpdn
+ p1 p2 p3 p4 p5 p6 p7 p8
+ PowerModel

* CPM Port Name | Average current (A) | Max. magnitude of current | Net voltage *
* p1 0.0483909 0.352188 1.1
. . .
* p8 -0.0396821 0.291764 0
* Average power = 0.21358 W.

Icursig1 p1 0 pwl(
+ 0.000000ps      0.13978214
+ 29.999999ps      0.15121435
+ 59.999998ps       0.1353907
+ 90.000001ps      0.14537421
```

Nx, Ny partition set in gv_args

PDN subckt included

CPP Mapping Info

CPP: chip package protocol

create a subckt, where Xpdn is the content

CPM subckt

instantiate one instance of PowerModel subckt as "Xpdn"

PDN instantiation

vdd port name (p1), gnd (0)
Each port has a current signature defined like this

Current signature

.end

Current Spice : Chip_Power_Model.sp

Command to get the CPM Output files

- <GrmView>.get_get_reduced_model_perform_ac_spice()
  - Returns the perform_ac.spice file

calculate input imped. btw vdd & vss

```
gv.get_reduced_model_perform_ac_spice()
```

```
* Calculate impedance(Zin) between nets VDD and VSS
* by the AC analysis of the passive part of the CPM model.
* For use with nspice simulator. Real and imaginary parts of Zin are plotted as function of freq
* For RC grids Rdie(freq) and Cdie(freq) plots also created
.include "Chip_Power_Model_SubCircuit.sp.inc"
Xdie VDD VDD VSS VDD VSS
+ VDD VSS VSS  PowerModel
Rlarge1 VDD 0 1e9
Rlarge2 VSS 0 1e9
Isrc  VSS VDD AC 1.0
.ac dec 50 5e7 2.5e9
* Set the frequency you want to calculate impedance at below:
.param freq=50e6
.meas ac real_v find vr(VDD, VSS) at=freq
.meas ac imag_v find vi(VDD, VSS) at=freq
.meas ac Re_Zin param='real_v'
.meas ac Im_Zin param='imag_v'
.option probe post
.probe Im_Zin_freq=par('vi(VDD, VSS)')
.probe Re_Zin_freq=par('vr(VDD, VSS)')
.probe Cdie_freq=par('-1.0/(2*3.14159265*hertz*vi(VDD, VSS))')
.probe Rdie_freq=par('vr(VDD, VSS)')
.end
```

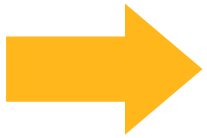
instantiate one instance of PowerModel subckt, as "Xdie"

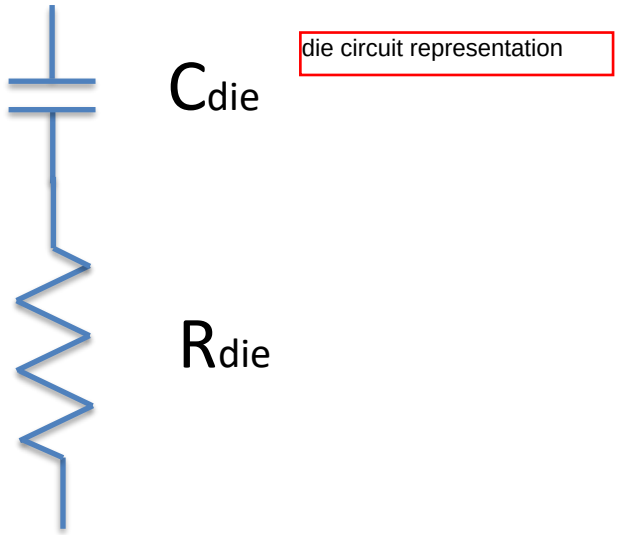
current source

AC freq analysis

perform_ac.sp

Effective RC Parameters from CPM





die circuit representation

$$Zin = R_{die} + 1/(j\omega C_{die})$$

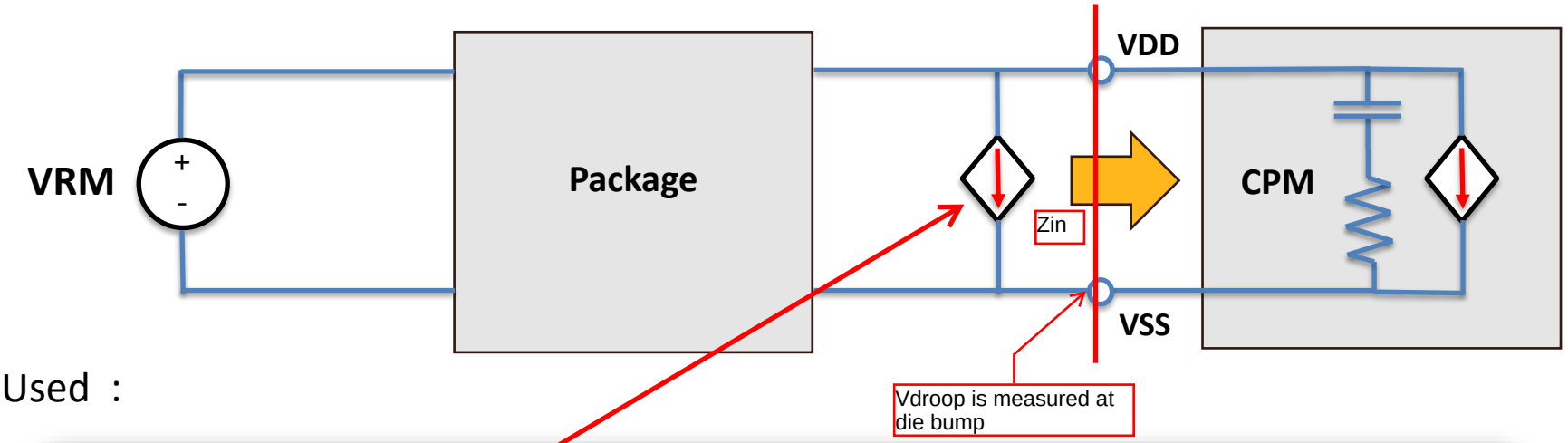
Cdie , Rdie vs Frequency Curve for Un-Coupled CPM

Cdie , Rdie vs Frequency Curve for Coupled CPM

Cdie Vs Frequency

Rdie Vs Frequency

Analyzing System Response using CPM



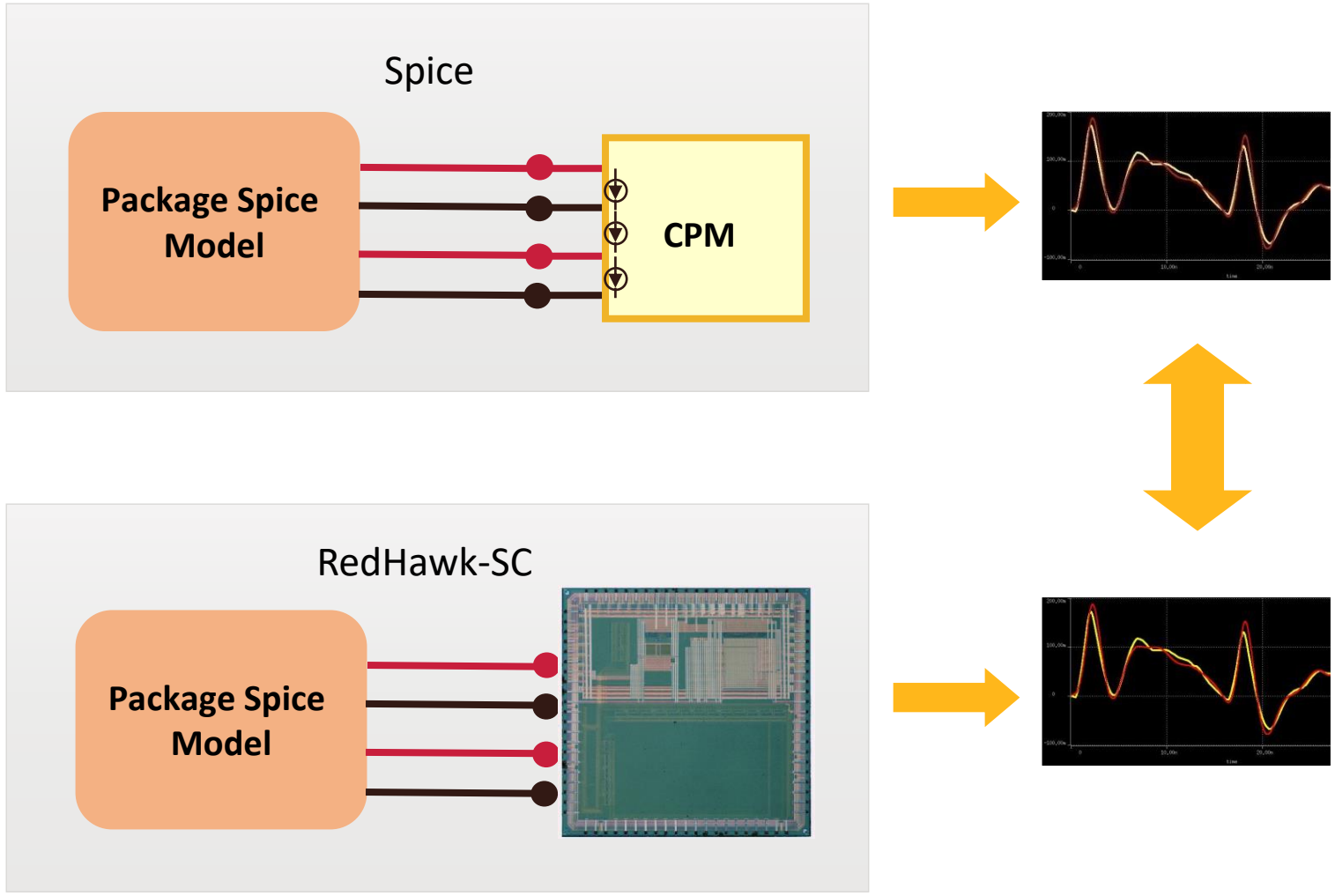
Spice Netlist Used :

```
* Calculate impedance(Zin) between nets VDD and VSS
.include "CPM.inc"
Xdie VDD VSS PowerModel

.inc 'redhawk_wrap_0p85v.sp'
Xpkg VDD VSS REDHAWK_PKG

Isrc VDD VSS AC 1.0
.ac dec 50 1e6 5e9

.option probe post
.probe Zin_freq=par('sqrt((vi(VDD,VSS)*vi(VDD,VSS))+(vr(VDD,VSS)*vr(VDD,VSS)))')
.probe Im_Zin_freq=par('vi(VDD,VSS)')
.probe Re_Zin_freq=par('vr(VDD,VSS)')
.end
```

CPM Hookup

Package Hookup

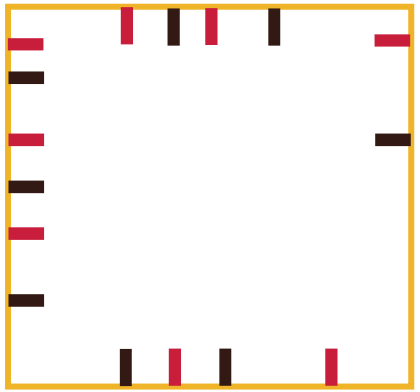
Self-Consistency Check for CPM

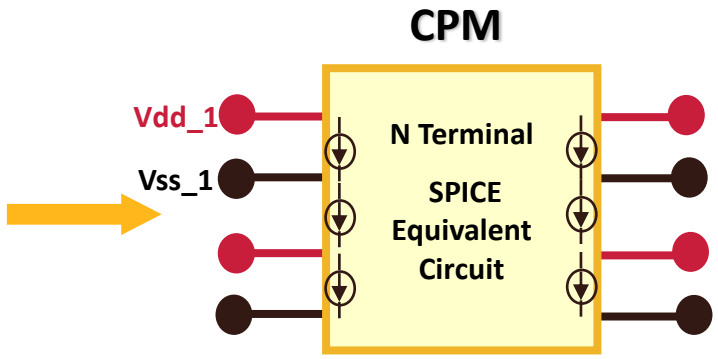


Additional CPM Configuration Options

CPM Port Creation Procedure

Wirebond chip





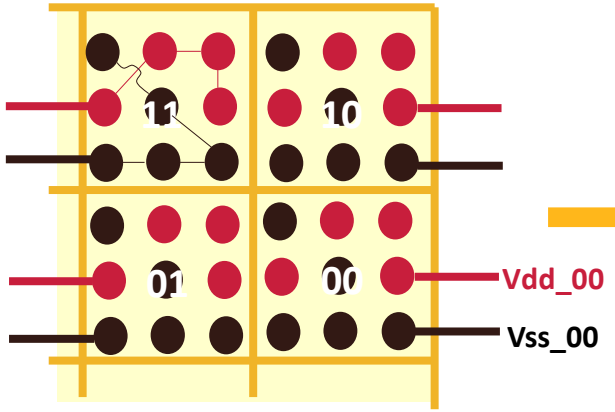
N = # of Partitions

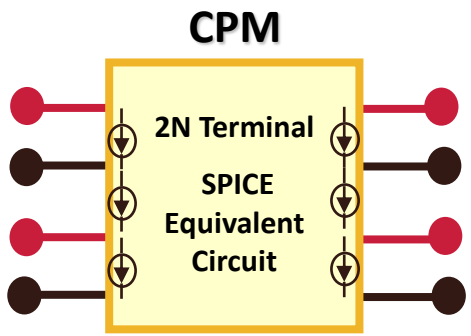
Default

each partition has one vdd, one vss terminal

N partition will have N vdd terminal, N vss terminal

Flip Chip Partitions





N = # of Partitions

for this exmaple, 4 paritions, 4x vdd, 4x vss, 8 terminals in total

Arguments to pass in
create_reduced_model_view
cpm_nx = <num_x_partitions> ,
cpm_ny = <num_y_partitions>

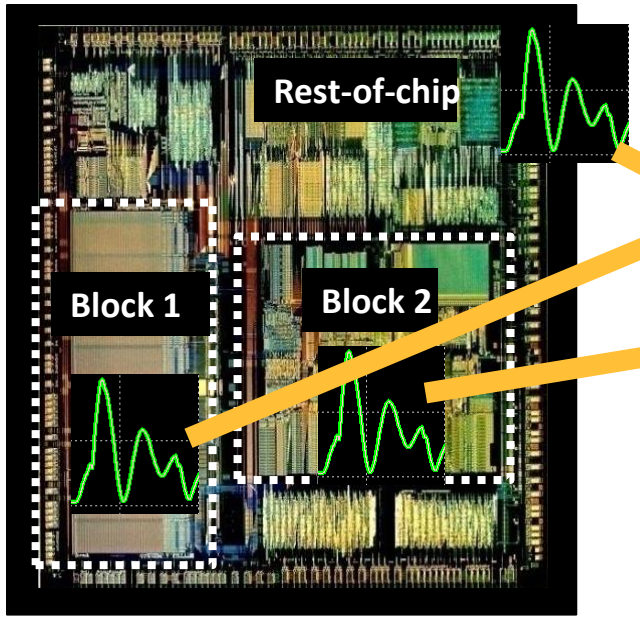
CPM Port Grouping - User Configurable via 6th Column in Ploc file

- Enables package and board designers to divide and customize a CPM for multiple individual system simulations
- Each targeting a specific operating mode or area of the chip
- Current signature is captured for each user-defined group
- Useful for examining impact of different blocks of the chip and their combined activity for DvD/EMI debug

```
#source name #loc_x #loc_y #layer name #P
DVDD1 10 5 METAL6 POWER GROUP_POWER_1
DVDD2 15 5 METAL6 POWER GROUP_POWER_1
DVDD3 10 5 METAL6 POWER POWER_2
DVDD4 15 10 METAL6 POWER POWER_3
DVSS1 10 15 METAL6 GROUND GROUP_GROUND_1
DVSS2 15 15 METAL6 GROUND GROUP_GROUND_1
DVSS3 10 5 METAL6 GROUND GROUND_2
DVSS4 5 5 METAL6 GROUND GROUND_3
```

6th Column in Ploc file specifies the group

Group definition



User-configurable CPM

```
I_group1_cursig p1 p2 pwl(
+ 0.000000ps 0.181292
....
+)
I_group2_cursig p1 p2 pwl(
+ 0.000000ps 0.181292
...
+)
I_others_cursig p1 p2 pwl(
+ 0.000000ps 0.155827
...
+)
```

CPM Port Grouping

- To enable Power/Ground bump grouping on different layers and regions

```
Synatx of <group_cfg_file> :
CPM_Port_Grouping {
#region_name layer nx ny llx lly urx ury_in_um
....
}
```

Passed via cpm_port_grouping_file

argument in create_reduced_model_view

```
Example:
CPM_Port_Grouping {
#region_name layer nx ny llx lly urx ury_in_um
DIE1_G PM0 20 20 3000.00 -4700.00 10000.00 4800.00
DIE2_G PM0 30 30 3000.00 -16100.00 10000.00 -6400.00
BGA_G UBM 20 40 -2500.00 -30000.00 15000.00 5000.00
}
```

Available Options In CPM Creation

Arguments	Description
cpm_cdie	Connects all ports of same net together to obtain a single-port solution to obtain the equivalent Cdie and Rdie value for chip.
cpm_plocname	Specifies the use of pad/group names for CPM port names.
cpm_parasitics	Generates only the passive part of the CPM
cpm_noglobal_gnd	Specifies the type of parasitic modeling in CPM without Spice Node 0
cpm_probes	Expose user named probes as external ports
cpm_nx	No of port groups along x direction
cpm_ny	No of port groups along y direction

*For more details, refer to help(SeaScapeDB.create_reduced_model_view)

CPM options file

- The Range of Frequency domain analysis by default starts from 10MHz to 2.5e+09 Hz, using Adaptive Frequency Sweep. If a design needs 10 GHz behavior the default can be changed using options file which can be passed to RHSC using argument cpm_options_file

```
# specifies the lowest frequency when using log grid default is 10MHz
fmin=1e3

# specifies the highest frequency for AC simulation default is 2.5GHz
fmax=10.0e9
```

Thank You

Ansys