Table of Contents

1. Introduction

For accurate dynamic voltage drop analysis in RedHawk, it is very important to have package parasitics (Resistance, Inductance and Capacitance) annotated in simulation. Although the resistance of the package is quite small, the inductance of package is significant which causes a voltage drop at the pad locations due to the time varying current drawn by the devices on die. Thus, the final voltage drop seen by the instances on die includes the package voltage drop and the IR-drop.

Excessive voltage drops in the power grid reduce switching speeds and noise margins of circuits, and inject noise which might lead to functional failures of the design. Hence it becomes critical to include the package effects for accurate dynamic voltage drop analysis in RedHawk.

Package parasitic include R, L and C values associated with Package, Bond-wire and Pad-wire segments. A simple package representation in RedHawk is shown here:

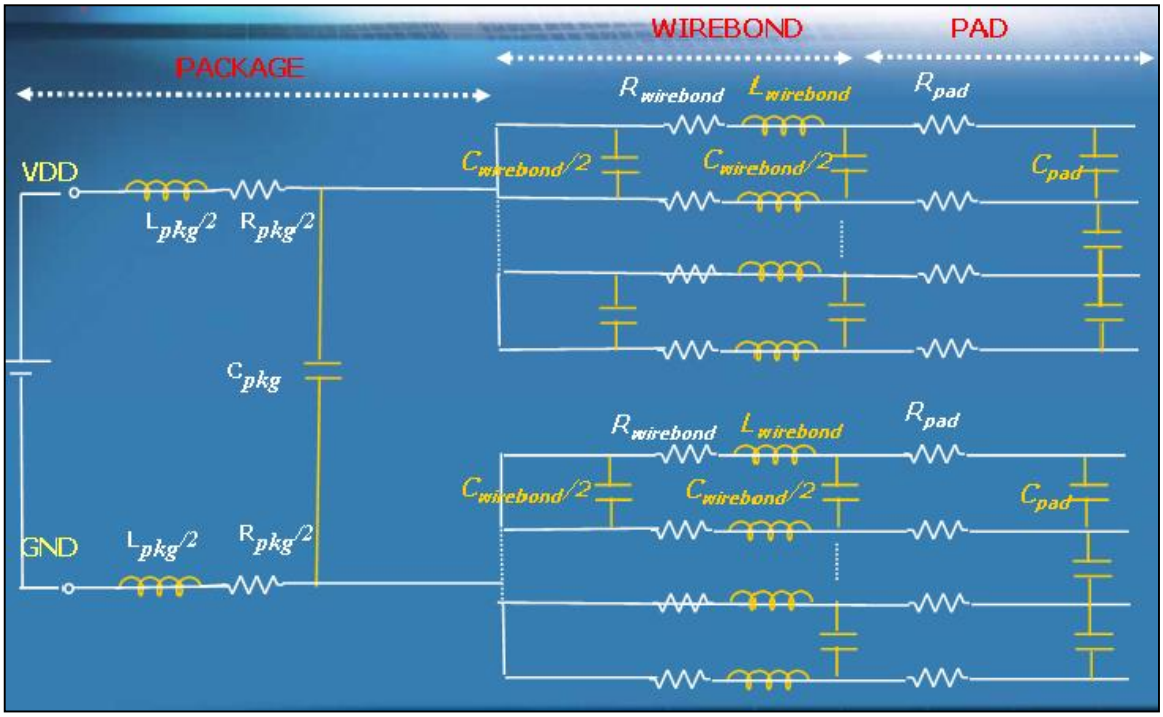

Figure 1 Package Representation in RedHawk

The $R_{wirebond}/L_{wirebond}/C_{wirebond}$ above represents the Bond-wire (Lead frame to bond-pad connection) parasitic. For Flip-chip designs, where there is no lead frame, this can be included as a part of package parasitic. R_{pad}/C_{pad} represent the parasitic of the Pad-wire (or bump-wire) routing (bond-pad/bump to IO pad cell connection). If this routing is already included in the design DEF, RedHawk will extract this automatically and you don't need to specify this value. $L_{pkg}/R_{pkg}/C_{pkg}$ represents the package layout parasitics.

2. Package Model Annotation

There are various ways for annotating package parasitic in the design. You can annotate each domain of the design with a simple lumped model or by hooking up a detailed RLC package Spice netlist to the die pads. Details of both methods are described below:

2.1 Using a simple lumped package Model

We can annotate a single lumped RLC value for power/ground domains in the design using RedHawk Tcl commands. Lumped model means that all pads belonging to a domain are shorted together and connected through the lumped RLC value to the VRM.

Here is the list of commands used for annotating simple RLC values:

```
setup package -power/ground -r <in Ohm> -c <in pF> -l <in pH>

setup wirebond -power/-ground -r <in Ohm> -c <in pF> -l <in pH>

setup pad -power/-ground -r <in Ohm> -c <in pF>
```

As you can see, wire-bond and pad constraints can be annotated separately for power and ground. Also there is no inductance value annotated with pad parasitic, as this routing is significantly small, and considered as zero in simulation.

Here is an example showing the above annotation:

```
setup pad -power -r 0.010000 -c 0.5

setup pad -ground -r 0.010000 -c 0.5

setup wirebond -power -r 0.245000 -l 1420 -c 5

setup wirebond -ground -r 0.245000 -l 1420 -c 5

setup package -power -r 0.001000 -l 10 -c 10

setup package -ground -r 0.001000 -l 10 -c 10
```

2.2 Using Package Spice Netlist

In this method, you define the detailed RLC of the package in the form of a spice sub-circuit and we apply the ideal voltage at the package balls through the sub-circuit. RedHawk will perform on-chip power integrity analysis with the effect of package spice netlist included in the simulation.

2.2.1 Defining the Package RLC and Ideal Voltage Sources

You need to include the package parasitic in the form of a spice sub-circuit. Following is an example for a package spice netlist. This sub-circuit location has to be defined in the GSR like this:

```
* Package subckt for design_abc.

.SUBCKT REDHAWK_PKG DIE_VSS1 DIE_VSS2 DIE_VDD1 DIE_VDD2
*
* Pin # 1 data
*
R1              DIE_VSS1            PM1                       0.18507
L1              PM1                 PE1                         1E-6
C1_0a           DIE_VSS1            0                      2.701e-014
C1_0b           PE1                 0                      2.701e-014
R1_l            PE1                 PM1_l                  8.53118E-02
L1_l            PM1_l               BGA_VSS1                    1E-6
C1_l            BGA_VSS1                 0                 3.74054E-13
*
* Pin # 2 data
*
R9              DIE_VSS2            PM9                       0.16643
RL9             PM9                 PE9                         1E-6
C9_0a           DIE_VSS2            0                     1.6675e-014
C9_0b           PE9                 0                     1.6675e-014
R9_l            PE9                 PM9_l                  8.20464E-02
L9_l            PM9_l               BGA_VSS2                    1E-6
C9_l            BGA_VSS2            0                     1.57874E-13
*
* Pin # 3 data
*
R24             DIE_VDD1            PM24                       0.1598
RL24            PM24                PE24                         1e-6
C24_0a          DIE_VDD1            0                     1.6052e-014
C24_0b          PE24                0                     1.6052e-014
R24_l           PE24                PM24_l                 7.86724E-02
L24_l           PM24_l              BGA_VDD1                    1E-6
```

```
C24_1            BGA_VDD1            0                    1.35616E-13
*
* Pin # 4 data
*
R20              DIE_VDD2            PM20                     0.12509
RL20             PM20                PE20                       1e-6
C20_0a           DIE_VDD2            0                   5.1465e-015
C20_0b           PE20                0                   5.1465e-015
R20_1            PE20                PM20_1              8.59898E-02
L20_1            PM20_1              BGA_VDD2                   1E-6
C20_1            BGA_VDD2            0                   3.71577E-13


* Supply node definition

VBGA_VSS1 BGA_VSS1 0 0
VBGA_VSS2 BGA_VSS2 0 0
VBGA_VDD1 BGA_VDD1 0 1.2
VBGA_VDD2 BGA_VDD2 0 1.2

.ENDS
```

.SUBCKT section defines the sub-circuit name and the nodes of the package Model that will be connected to the die.

RedHawk requires that sub-circuit should be defined by the name "REDHAWK_PKG". The nodes defined (DIE_VSS1, DIE_VSS2 etc.) in "REDHAWK_PKG" sub-circuit are the die-side terminals of the package spice netlist which in turn have to be connected to the pad-locations (This connection is not defined through this spice deck. It has to be defined in the pad location file, which is explained later). .ENDS specifies the end of the sub-circuit.

Nodes names BGA_VSS1, BGA_VSS2 etc. are the battery side connections. We are applying ideal voltage sources in the supply nodes section like this.

```
VBGA_VSS1 BGA_VSS1 0 0
VBGA_VSS2 BGA_VSS2 0 0
VBGA_VDD1 BGA_VDD1 0 1.2
VBGA_VDD2 BGA_VDD2 0 1.6
```

Note that, nodes corresponding to a ground pad should be connected to 0V and nodes connected to power pads should be connected to its ideal supply voltage (1.2V here).

* Pin # section defines the parasitic associated with each pad in the design.

```
R1                  DIE_VSS1            PM1                      0.18507

L1                  PM1                 PE1                        1e-6

C1_0a               DIE_VSS1            0                    2.701e-014

C1 0b               PE1                 0                    2.701e-014
```

You can define the RLC values here. RedHawk supports regular SPICE constructs like R/L/C/K/E/F/G/H elements in the package spice netlist.

Use the keyword PACKAGE_SPICE_SUBCKT <Spice netlist file> to include the above defined Spice model in your analysis.

2.2.2 Connecting Package Sub-circuit to the Die

We need to define the package to die connection through the PLOC file. PLOC file contains the bond pad locations, with corresponding net name and the metal layer information. Here is an example showing the format of this file.

```
#source name #loc_x   #loc_y   #layer-name #P/G

BONDPAD_1    2377.45  4514.525 MET6         VSS

BONDPAD_2    3464.225 4514.525 MET6         VSS
```

You need to define the package connection associated with each pad in the above file. Format is like this:

```
<pad_name> <X> <Y> <layer> <DOMAIN> <package node>
```

Here is an example showing the modified PLOC file.The package nodes have been highlighted:

```
 BONDPAD_1 2377.45 4514.525 MET6 VSS DIE_VSS1

 BONDPAD_2 3464.225 4514.525 MET6 VSS DIE_VSS2

 BONDPAD_3 2116.25 1836.9 MET6 VDD DIE_VDD1

 BONDPAD_4 16.25 2855.325 MET6 VDDR DIE_VDD2
```

You need to define the above “annotated” PLOC file in your GSR

```
PAD_FILES {
design.package.ploc
}
```

3. Package Modeling using RedHawk-CPA

RedHawk-CPA brings an integrated GUI to RedHawk users for chip-package analysis in a seamless flow. In RedHawk GUI itself, user can explore, edit and model the package design conveniently. Generating package model and importing into RedHawk is a “one-click” operation thanks to the automated design setup and the natively created interface files. RedHawk-CPA provides TCL commands that incorporate package model preparing step into the existing RedHawk flows. In addition to the integrated GUI, RedHawk-CPA provides per-bump resolution modeling capability and high performance analysis solution.

Key features:

- Integrated GUI that shows chip and package layout / map views together
- Enables package layout import and CPA model generation through the RedHawk GUI
- Fully distributed, chip analysis ready, per bump parasitic network
- Automatic hook-up to chip layout maintaining pin-to-pin mapping
- Support TCL commands

- Layout manager for package layout viewing and editing
- 3D Fast Finite Element Method engine

3.1 GSR Keywords

Use the keyword CPA_FILES in GSR for package modeling using RedHawk-CPA.

Syntax:

```
CPA_FILES {
        PACKAGE <package_layout_filename>
        MODEL <project_path>
}
```

- **PACKAGE**: specifies the package layout file to be displayed and imported into RedHawk-CPA. The supported formats are xfl, mcm, sip and brd. The layout file name is used as the project name.
- **MODEL**: defines the path to CPA project directory. A new directory will be created if the model does not exist. If the MODEL keyword points to a previously extracted package model, the existing model will be loaded, and used for RedHawk voltage drop simulation.

Usage example as to be used in the GSR:

```
CPA_FILES {
        PACKAGE analysis.xfl
        MODEL ./model_cpa
}
```

MCM, SIP, and BRD file formats require CDSROOT environment variable for successful import into RedHawk.

3.2 Package GUI and chip+package co-visualization

In order to view/edit package layout and perform extraction, you can start the integrated RedHawk-CPA GUI in the following step:

Define `CPA_FILES` keywords in GSR file.

Create a command file that contains:

```
import gsr <GSR_file>
setup design
```

Run RedHawk from GUI by executing the command file that created in step 2:

```
%redhawk -f <command_file>
```

View the package layout by clicking the PKG radio button as shown in Figure 3. Alternatively, view the package and chip side by side by selecting multiple pages:

Windows > Multiple Pages > Setup

Select the Package and Chip layout to view the package and chip GUI simultaneously as shown in Figure 2

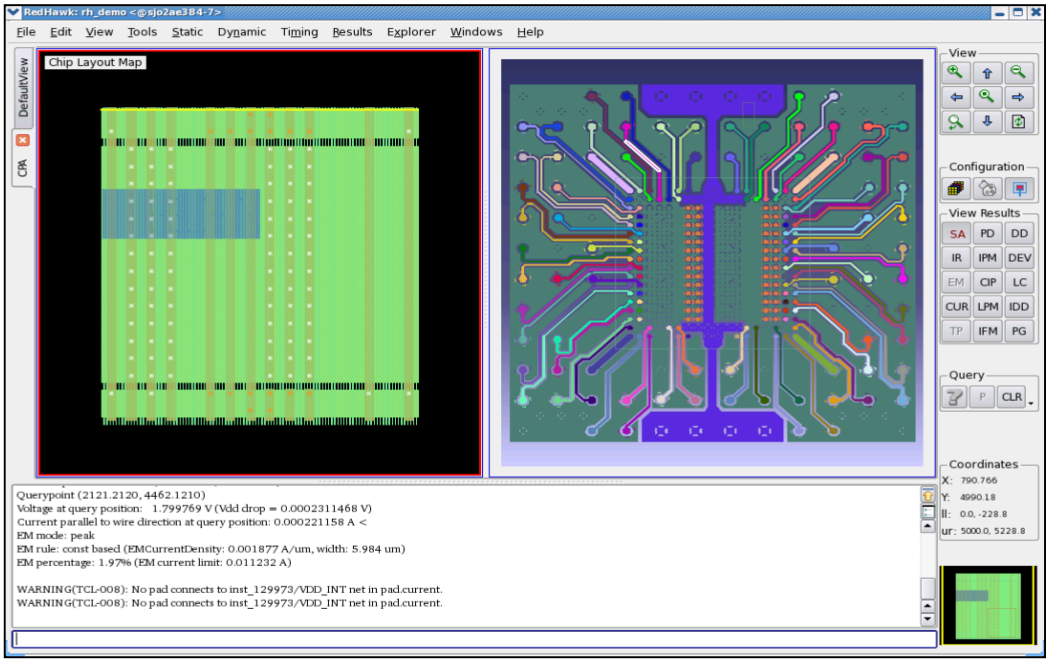

Figure 2 : Integrated RedHawk-CPA GUI showing package and chip layouts simultaneously

3.3 TCL Command for extracting the package layout

Perform the automatic setup of the package using the following Tcl command:

- `setup cpa auto`

This Tcl command performs the automatic setup that:

1. Perform automatic ploc connection using the ploc file specified by PAD_FILES in GSR.

2. Select the P/G nets specified in GSR for extraction.

3. Setup VRM with the voltage specified in GSR.

Perform the extraction of the package once the design/package setup is complete using the following Tcl command:

- `perform cpa extraction`

This package extraction command does the following:

1. Save project files to <project_path> specified by MODEL keyword
2. Perform DRC check;
3. Generate package model and interface files that connects package model to RedHawk

3.4 Viewing the Package Extraction Results

Various maps depicting the resistance and inductance of a package pin as seen by the die are available after package extraction is complete. Per-Bump resolution of Pin R and Pin L can be viewed in the result maps as shown below:

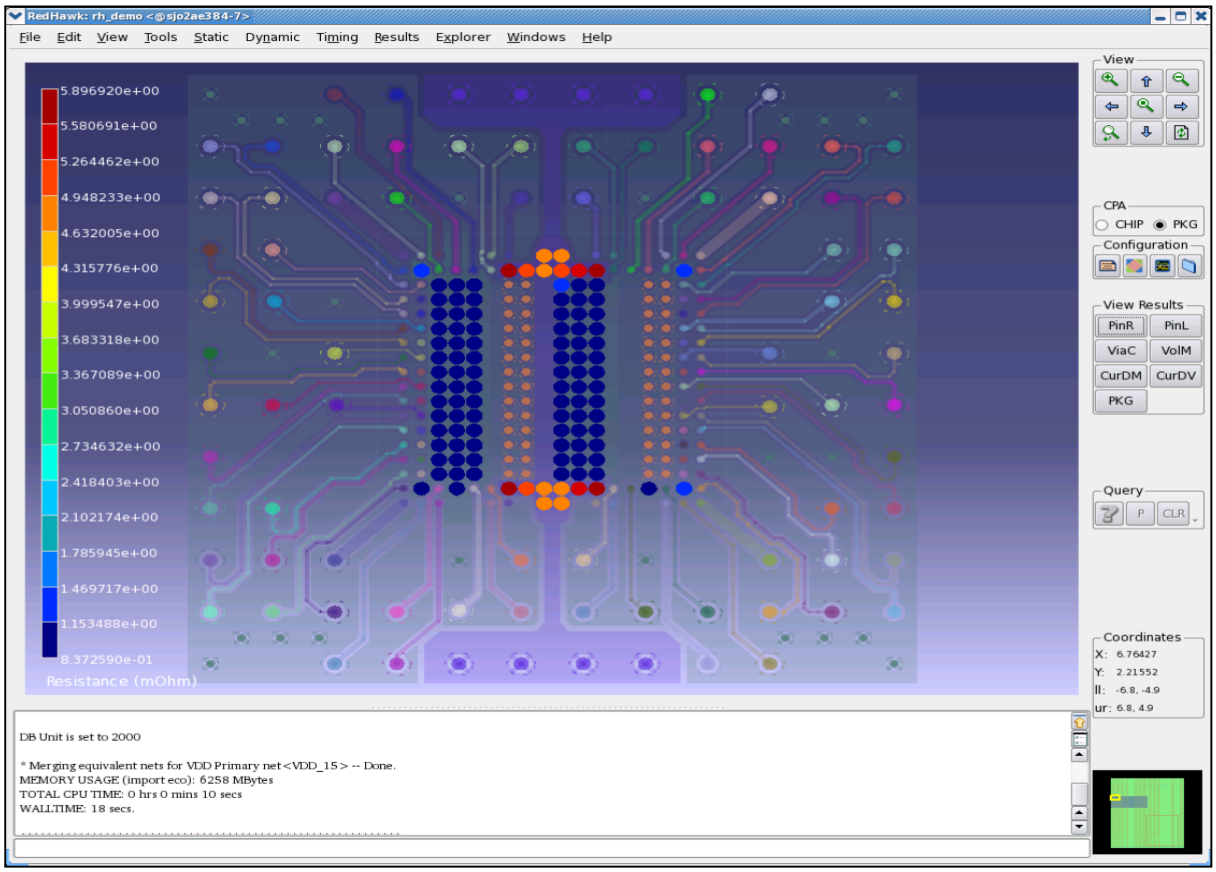

Figure 3 : Integrated RedHawk-CPA GUI showing package pin R values as seen by the chip

Text Reports: The generated package model is located in <project_path>/Extraction folder:

File name	Description
<job_name>.sp	RLCK Spice model of the package
<job_name>_<component>.sp	Subckt model of the components with ports
<job_name>_<component>.cpp	CPP header specifies components with ports
cpa_annotated.ploc	Annotated PLOC file (encrypted)
cpa_rh_pkg_wrapper.sp	Top level netlist for RedHawk (encrypted)
unconnected_plocs.txt	Lists the ploc pins that are not used or floating for PLOC connection
Lumped_L.txt	Effective inductance for each domain

Partial_L.txt	Partial inductance for each package pin
Chip_PKG_NetMap.txt	Chip->Pkg net connections

Table 1 RedHawk-CPA Generated Text Reports

3.5 Performing Package Aware Static/Dynamic/Rampup Analyses

Analysis can be done via regular “perform analysis” commands – RedHawk will automatically include the extracted package into the simulation.

Tcl Syntax:

```
perform cpa extraction

perform analysis <-dynamic | -static | -lowpower>
```

You can view the Pad Voltage map and the Package RL map simultaneously as shown:

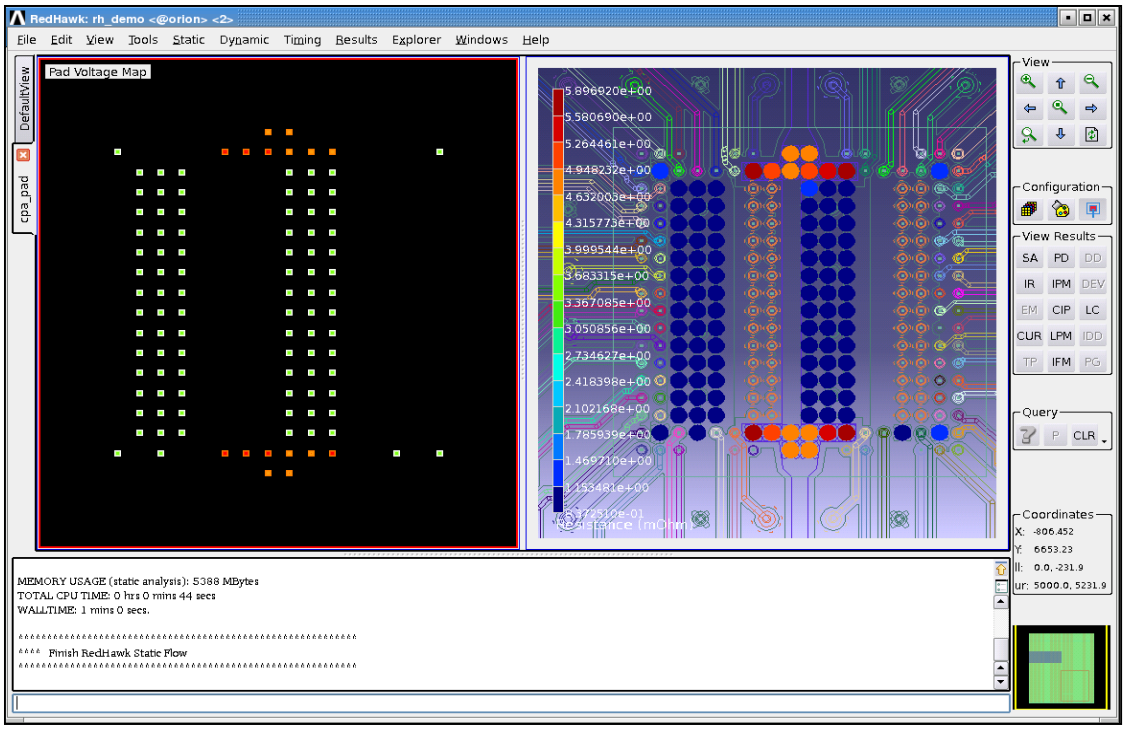

Figure 4 Pad Voltage Map and Package Pin RL Map

3.5.1 Viewing Package drop contribution in RedHawk- Explorer

You can review the Summary>Static/Dynamic Voltage Drop Summary to see the voltage drop contribution from chip and package in RedHawk-Explorer as shown in Fig4

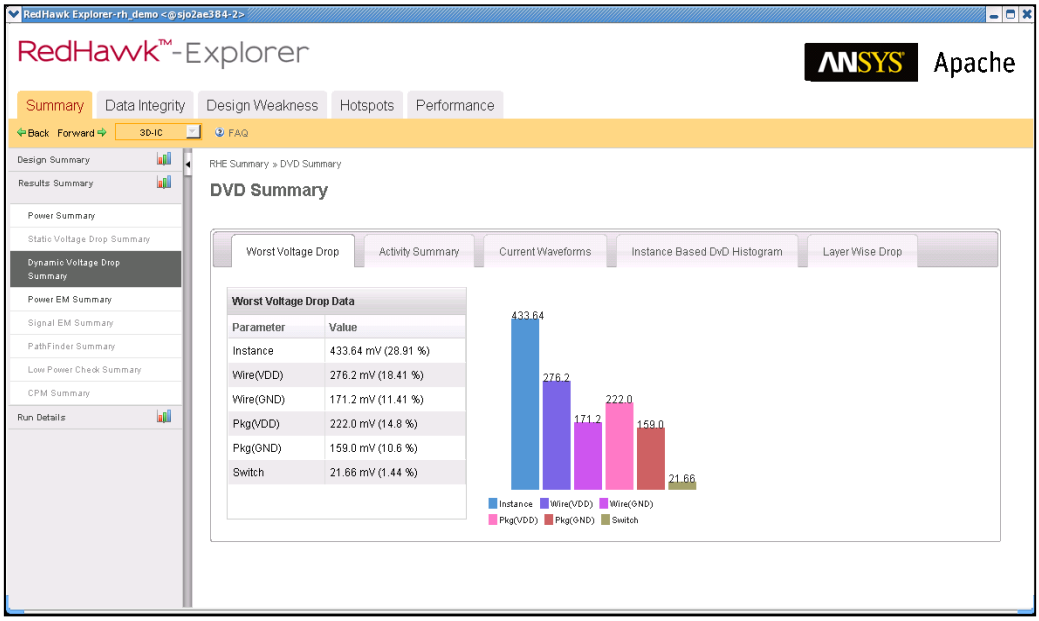

Figure 5 Histogram showing the Dynamic Drop Contribution

You can review 'Data Integrity> Package Settings' page to make sure the package is included in the simulation as shown in Figure 5

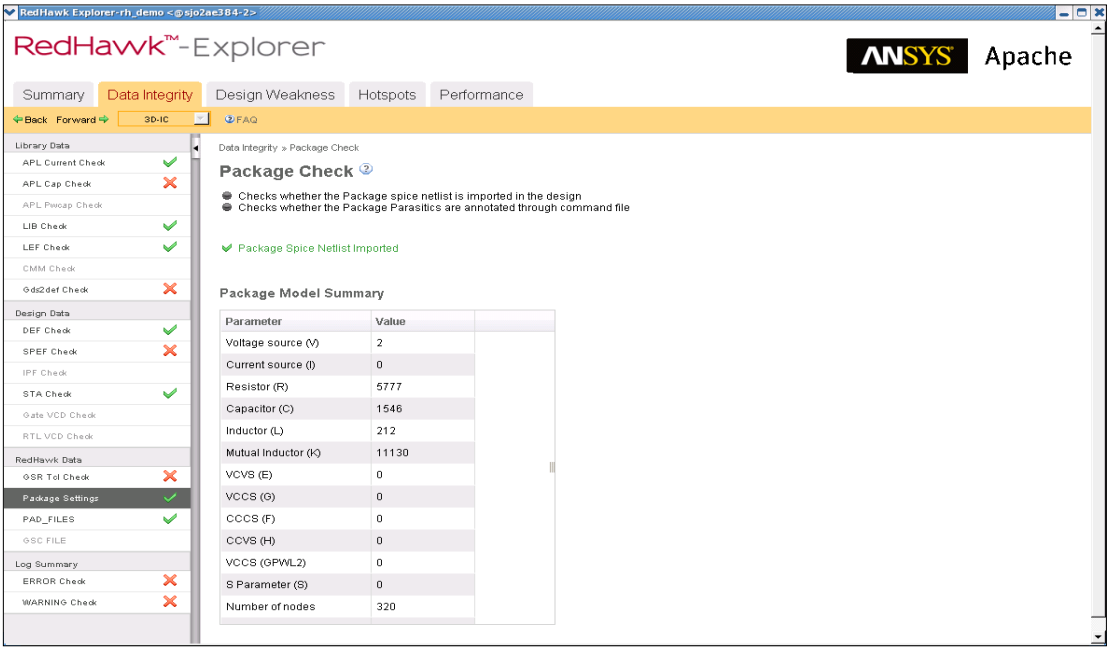

Figure 6 Package Model Summary page in RedHawk

You can review the drop contribution from the package for both power and ground domains under Design Weakness>Package Drop Contribution as shown in Fig 7

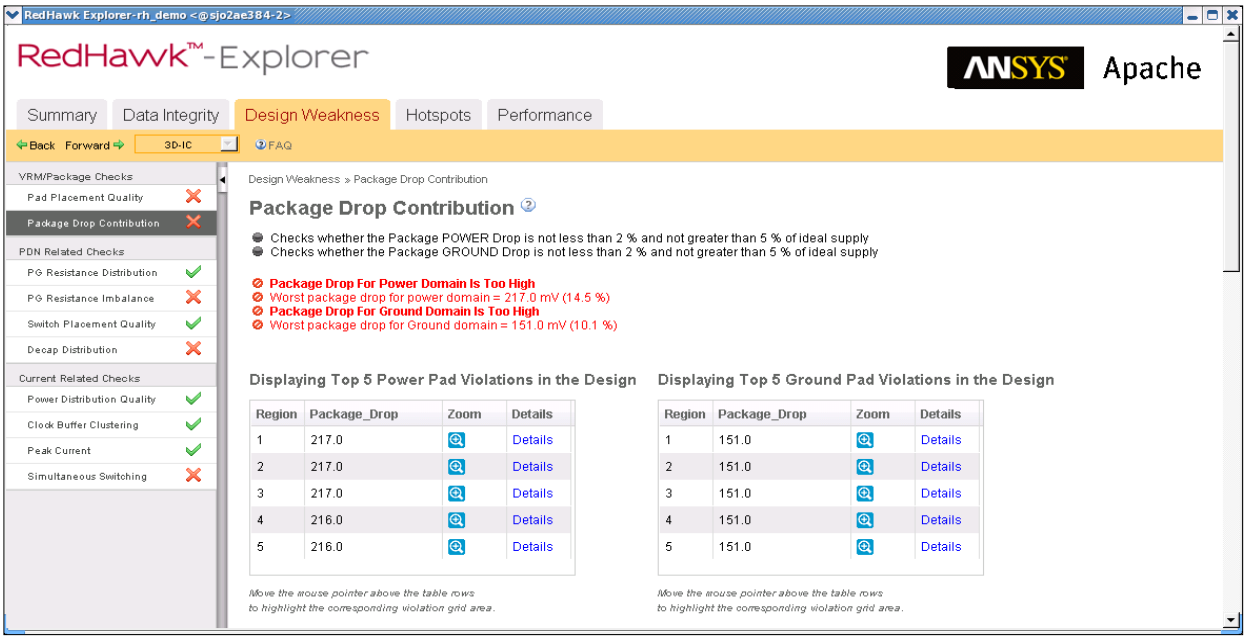

Figure 7 Package Model Summary page in RedHawk

3.6 S-Parameter PCB Modeling

You can specify the PCB S-Parameter Model while doing the static/dynamic/ramp up analysis in RedHawk. Connection and setup of the model is similar to RLCK package Model annotation as shown previously.

The PCB S-parameter Model can also be included in the Package Model extracted by RedHawk-CPA by modifying the <project_name>_BGA.sp file.

For using the S-Parameter PCB Model along with the RedHawk-CPA package model in RedHawk analysis, modify the <project_path>/Extraction/<project_name>_BGA.sp file as shown below.

```
.subckt fccsp_BGA
+ Node_BGA_VSS Node_BGA_VDD

*Instantiate the S-Parameter PCB Model to Package Nodes port
Npkg S(2)
+ Node_BGA_VSS Node_BGA_VDD
+ pcbVdd pcbVss PCB_MODEL

*Include the PCB_MODEL S-parameter file
.model PCB_MODEL nport file = "pkg.s2p" np = 2

*Hook up the Supply Source on the PCB BGA Nodes
V1 pcbVdd 0 1.000000e+00
V2 pcbVss 0 0.000000e+00
.ends
```

Note: The supply sources on the BGA side should be between a node and ground (SPICE node 0)

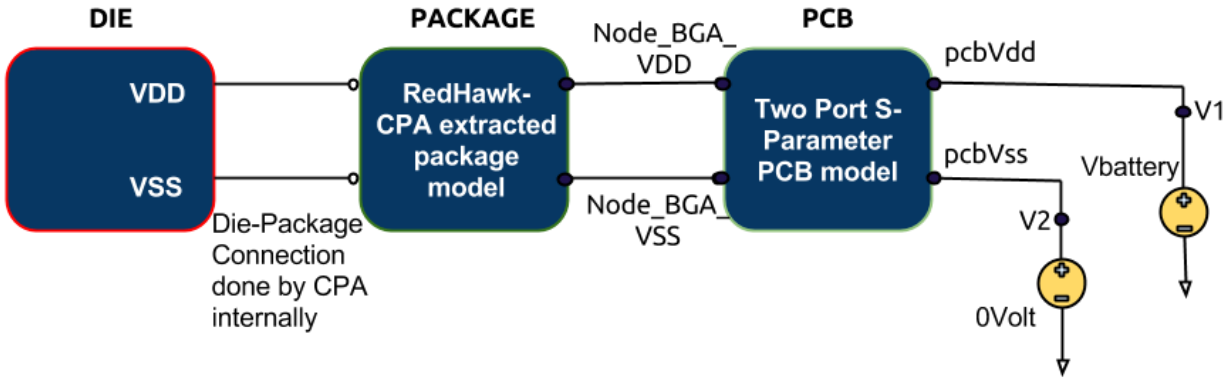

Figure 8 Including PCB S-Parameter in CPA extracted package Model

ANSYS, Inc.
2645 Zanker Road
San Jose, CA 95134

Main: 408-457-2000
Fax: 408-428-9569
redhawk_support@ansys.com
apache_sales@ansys.com
www.ansys.com
www.apache-da.com