Thermal Simulation Flow and Thermal Closure Methodology in High-Performance VLSI/SOC Design

QING K. ZHU
Electrical Engineering Department, International Technological University
355 W. San Fernando St., San Jose, CA 95113,
USA

Abstract: - This paper describes our thermal simulation flow and thermal closure methodology based on one high-performance VLSI/SOC chip. Thermal simulation results help the design team to evaluate the cooling technology and verify the highest temperature on the chip. The developed flow is based on PowerDC tool from Cadence Design Systems to automate steps based on one configure file. We discuss more details about the tool, the configure file and power density file that are critical to the accuracy of simulation results. The general thermal closure methodology and engineering teams involved in the process are described in the paper with experimental results based on the high-performance SOC chip.

Key-Words: Thermal simulation, temperature, power, cooling, SOC, VLSI, heat sinks, cold plates.

Received: May 27, 2020. Revised: October 6, 2020. Accepted: November 2, 2020. Published: November 17, 2020.

1 Introduction

SOC (System-on-Chip) and AI (Artificial Intelligence) semiconductor chips integrate architecture parallelism and functions in one chip. Semiconductor technology advance and high-performance requirement also drive the continuous integration with the cost of higher power consumption. Table I lists performance and power numbers in modern GPU (Graphical Processing Unit) [1]. The maximum power consumption in GPU chip can be close to 300W. Temperature or thermal management becomes challenging in semiconductor chips due to the increasing power consumption. Note that temperatures on the chip can influence circuit performance such as device threshold voltage, carrier mobility and circuit driving capability [2-3]. Detailed and accurate thermal simulation for the VLSI system including the die, package and cooling environment is necessary to get temperature distribution and the highest temperature on the chip before we deliver design database for IC manufacturing. We need to make sure the highest temperature on the chip is within the upper limit set by design team in circuit design specification. Figure 1 shows thermal model of a die mounted on package and heat sinks cooling environment. Power consumption p(t) of the die is represented by the current source. p(t) is a time and location dependent variable to model dynamic currents of circuits in the die. The analogy between thermal model and electrical model shown in Appendix I has been revealed by Roberson and Gross in the 1958's publication [4]. Junction temperatures between interfaces of die, package and heat sink are modelled similar to node voltages in the electrical model. Temperatures in thermal model can be solved based on nodal equations using liner matrix solution [5-6].

Table I. Performance and Power Numbers in GPU Chips.

CORE (MHZ)	MEMORY (MHZ)	PERFORTMANCE (GFLOPS)	POWER (WATTS)
550	1100	69.60	289.64
600	1000	71.01	290.57
650	900	69.70	298.36

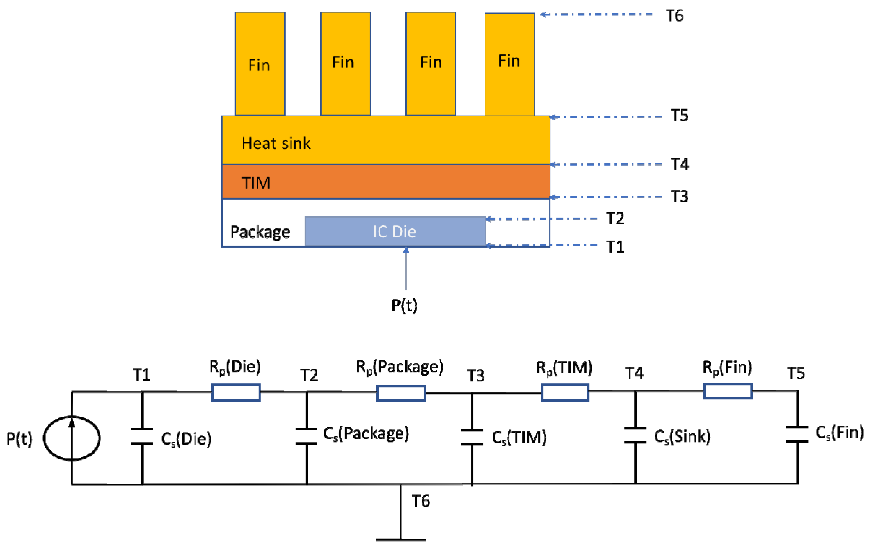


Figure 1: Thermal Model of IC Die Mounted on Package and Heat Sink.

Note that on-chip temperatures can influence metal resistances and device currents of circuits. The variation of device currents can go back to impact temperatures on the chip. Therefore, we need to couple thermal and electrical models together in the thermal simulation. This is called the thermal/electrical co-simulation as shown illustrated on Appendix II. Co-simulation or the dynamic loop between thermal and electrical models improves the simulation accuracy. One EDA tool PowerDC from Cadence Design Systems Inc adopts this co-simulation mechanism to simulate thermal effects on the chip based on the modeling of the die, package and cooling environment in the DC domain [7-8]. RedHawk and Totem tools from Ansys are based on FEM modeling and thermal/electrical co-simulation mechanism [9-10]. FloTHERM tool from Mentor Graphics is based on object-based meshing and novel modelling technique [11]. Many academic research publications can be found in the literature on thermal modelling and simulation tools. Traditional finite difference, finite-element or boundary element methods can be used to solve heat conduction equations with high accuracy but extremely long computational time in a complex VLSI system [12-14]. Green function in the 2-D model can be solved in reduced simulation time but less accurate result is achieved [15-16]. Cheng et al. developed the ILLIADS-T electrothermal timing simulator that reports steady-state temperature profile and identify reliability issues in the chip [17]. Chang et. al developed the GA2CO simulator that estimates the lower bound of peak temperatures in the chip including packages [18]. A compact model in the second-order polynomial curve fitting of transient simulation waveforms was developed to estimate on-chip temperatures [19].

This paper is organized in six sections. Section 2 discusses thermal closure and signoff concepts and methodology. Section 3 describes the flow and configure file we developed for to automate thermal simulation tasks based on Cadence PowerDC tool. Section 4 describes formats and construction methods of power density files in early design stage and final chip tapeout. Section 5 shows experimental results in one high-performance SOC chip. Section 6 gives conclusions. Three appendixes are attached in this paper to explain the analogue between electrical and thermal models, thermal/electrical co-simulation mechanism and TCL code to automate the thermal simulation flow.

2 Thermal Closure and Signoff

Temperatures on the chip can have significant impacts on the circuit speed and reliability. Design team usually defines temperature upper and lower bounds when designing and simulating the circuit. The lower and upper bounds of temperatures in the die specified by the design team has to be verified based on the thermal simulation. When the thermal simulation shows the highest temperature in the die is out of the required upper bound, these thermal violations need to be fixed by the design or optimize the cooling technology to remove more heats. For example, when design team specified 100C as the upper limit, simulated temperatures in the die above 100C are thermal violations or hot spots.

Here are terminologies in the thermal closure and signoff methodology.

- *Thermal violations or hot spots*: locations in the die where temperatures above the required temperature upper limit.
- *Thermal closure*: efforts to optimize the design and choose the cooling technology to remove all of thermal hot spots in the die.
- *Thermal signoff*: thermal closure and verification before the chip tapeout to make sure all the thermal violations are removed in the design.

Successful thermal closure in a complex VLSI chip design needs coordinated efforts from engineering teams. Figure 2 shows engineering teams involved in thermal management. Automation team develops thermal simulation flow and selects EDA tool for the accurate thermal simulation. Cooling technologists select vendors and work with the design team to select feasible cooling technology. Design team executes the thermal simulation and optimizes the architecture, circuits, floorplan and power distribution to remove thermal hot spots on the die. The detailed tasks involved in final thermal sign off process is shown in Figure 3. Note that thermal hot spots and removal need accurate thermal simulation results for design team to make decisions. We will discuss more details about the thermal simulation CAD flow in Section 3.

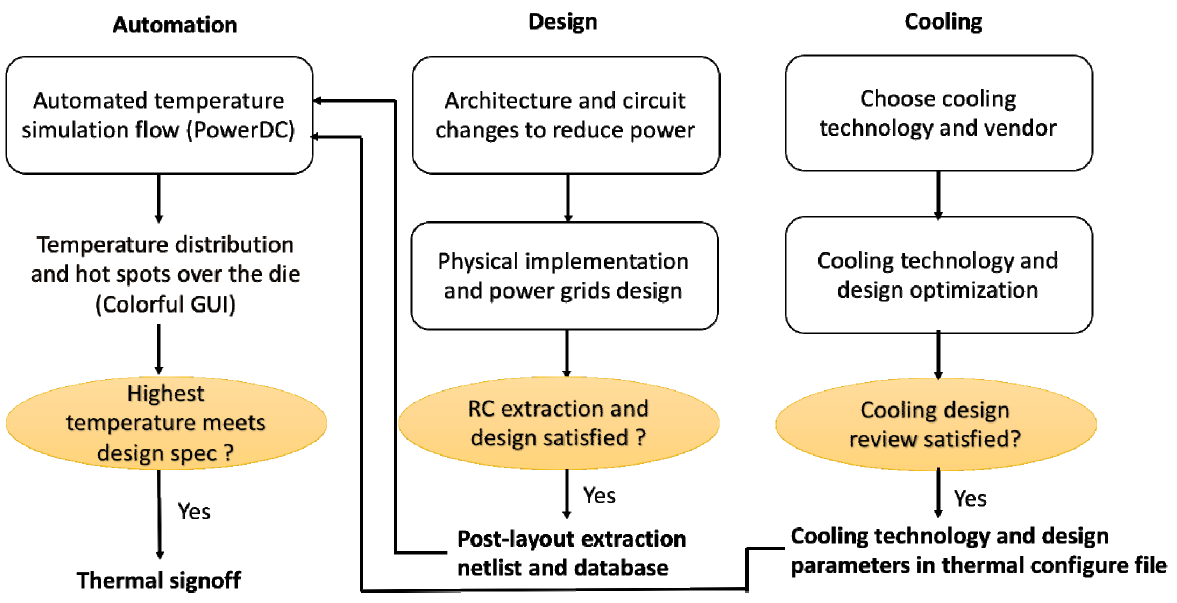

Automation
Design
Cooling
Automated temperature simulation flow (PowerDC)
Architecture and circuit changes to reduce power
Choose cooling technology and vendor
Temperature distribution and hot spots over the die (Colorful GUI)
Physical implementation and power grids design
Cooling technology and design optimization
Highest temperature meets design spec ?
RC extraction and design satisfied ?
Cooling design review satisfied?
Yes
Yes
Yes
Thermal signoff
Post-layout extraction netlist and database
Cooling technology and design parameters in thermal configure file

Figure 2: Engineering Teams Involved in Thermal Closure and Responsibilities.

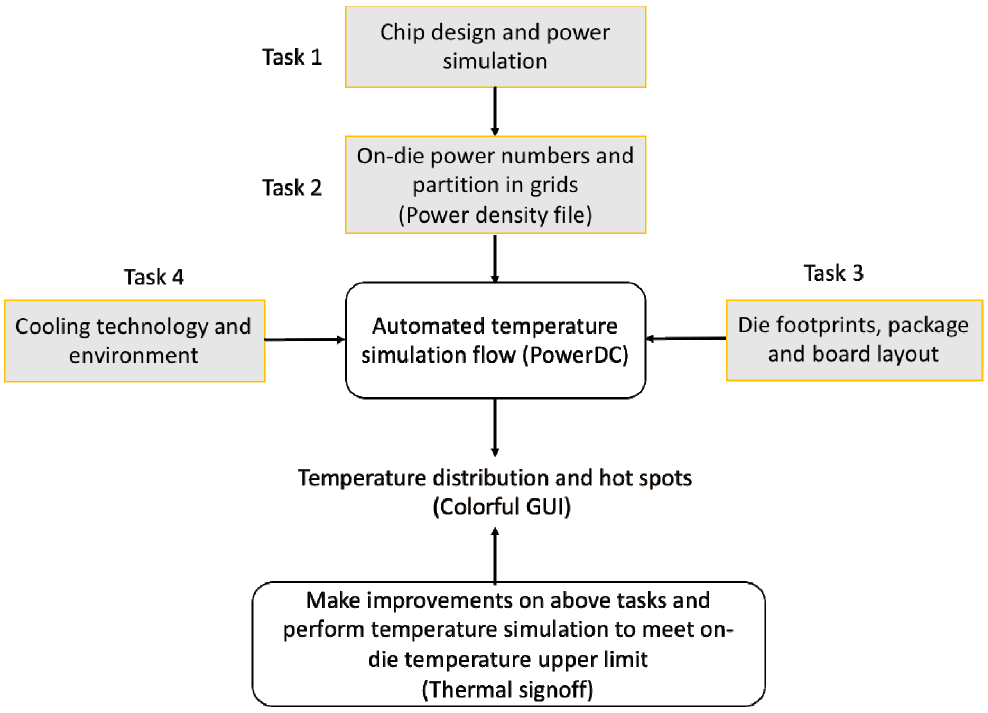

Task 1
Chip design and power simulation
Task 2
On-die power numbers and partition in grids (Power density file)
Task 4
Cooling technology and environment
Automated temperature simulation flow (PowerDC)
Task 3
Die footprints, package and board layout
Temperature distribution and hot spots (Colorful GUI)
Make improvements on above tasks and perform temperature simulation to meet on-die temperature upper limit (Thermal signoff)

Figure 3: Tasks in Thermal Sign-off.

3 Flow and Configure File

We developed one automation flow to prepare and execute the thermal simulation based on Cadence PowerDC tool. The flow is based on one configure file as shown in Figure 4. Table I lists three input files that are specified in the configure file. Other parameters in the configure file are described in Table II. It is significant effort to prepare input files that are among few engineering teams including the CAD, design team and cooling technologists as shown in Table III. We will discuss the power density file in Section 4.

```
# This is the config file to run thermal analysis flow.
wsfile <path1>/package_heat_sink.pdcx
layfile <path2>/package_layout.spd
pwrfile <path3>/powerMap_<data>.txt
temperature 55
airvelocity 1.5
tempfile    die_temps_heatSink.txt
# Other package material/structure parameters
name BGA_heatSink
type Fin
sinkmaterial aluminum
length 0.027
width 0.034
basethickness 0.002
limbheight 0.025
limbthickness 0.001
limbpitch 0.0325
fingerwidth 0.007
fingerperiod 0.014
diameter 0.0072
countl 2
countw 9
offsetl 0
offsetw 0
```

```
adhesivematerial tim_heatsink
adhesivethickness 0.0025
heattransfercoefficient 0

# Submit thermal analysis flow or not in PowerDC environment
run 1`
```

Figure 4: Configure File.

Table I. Input Files Specified in Thermal Simulation Configure File.

FILE	SPECIFICAITON
WSFILE: THERMAL WORKSPACE FILE	IT SPECIFIES THE WORKSPACE TO RUN THERMAL ANALYSIS. IT INCLUDES PACKAGE LAYERS AND MATERIALS, POWER NET NAMES AND SIMULATION SETUP CONDITION.
LAYFILE: LAYOUT DESIGN FILE	IT SPECIFIES THE PACKAGE LAYOUT'S METAL LINES, VIAS AND NETS. IT ALSO SPECIFIES COMPONENTS OR CHIPS MOUNTED ON THE PACKAGE.
PWRFILE: POWER DENSITY FILE	IT SPECIFIES POWER DENSITY DISTRIBUTION FOR EACH CHIP MOUNTED ON THE PACKAGE.

Table II. Thermal Parameters Specified in Configure File.

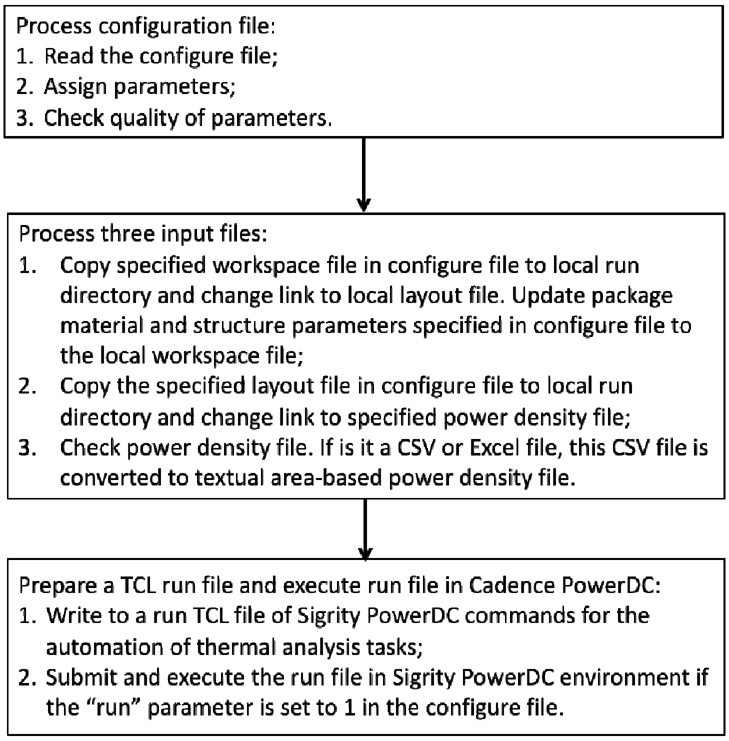
PARAMETER	SPECIFICAITON
TEMPERATURE	ROOM TEMPERATURE TO MODEL MATERIAL CONDUCTIVITY
AIRVELOCITY	RATE OF MOTION OF AIR IN GIVEN DIRECTION IN THE UNIT OF METERS PER SECOND
NAME	NAME OF PACKAGE AND COOLING STRUCTURE
TYPE	COOLING TECHNOLOGY CHOICE (HEAT SINK OR COLD PLATE)
SINKMATERIAL	MATERIAL TO HEAT SINK FINS
LENGTH	LENGH OF HEAT SINK STRCUTURE
WIDTH	WIDTH OF HEAT SINK STRUCTURE
BASETHICKNESS	THICKNESS OF HEAT SINK BASE
LIMBHEIGHT	HEIGHT OF LIMBS IN HEAT SINK
LIMBTHICKNESS	THICKNESS OF LIBMS IN HEAT SINK
LIMBPITCH	PITCH OF LAMBS IN HEAT SINK
FINGERWIDTH	WIDTH OF FINGERS IN HEAT SINK
FINGERPERIOD	PITCH OF FINGERS IN HEAT SINK
DIAMETER	DIMENSION OF FINGERS
COUNTL	NUMBER OF FINGERS IN LENGTH DIRECTION
COUNTW	NUMBER OF FINGERS IN WIDTH DIRECTION
ADHESIVEMATERIAL	MATERIAL TYPE OF ADHESIVE TIM
ADHESIVETHICKNESS	METRIAL THICKNESS OF ADHESIVE THERMAL INTERFACE (TIM)
HEATTRANSFERCOFFICIENT	COFFICIENT OF HEAT TRANSFER THROUGH ADHESIVE THERMAL INTERFACE

Table III. Teams and Tasks to Prepare Three Input Files.

INPUT FILE	TEAM	TASKS
WSFILE: THERMAL WORKSPACE FILE	CAD	SETUP WORKSPACE FOR THE PROJECT ON THERMAL ANALYSIS
LAYFILE: LAYOUT DESIGN FILE	PACKAGE LAYOUT	PACKAGE AND LAYOUT DESIGN
PWRFILE: POWER DENSITY FILE	CAD AND DESIGN	ARCHITECTURE OPTIMIZATION, LOGIC AND CIRCUIT DESIGNS, FLOORPLANNING, POWER DISTRIBUTION, POWER SIMULATION, GENERATION OF POWER PROFILES
CONFILE: CONFIGUE FILE	CAD AND COOLING ENGINEERING	COOLING CHOICES, COLLECTION AND REVIEWING OF PARAMETERS IN CONFIG FILE

The core simulation engine in the flow is PowerDC tool from Cadence Design Systems. This tool has the rich GUI that the design team usually runs based on the assistance from CAD engineers. But the specifications and flow in GUI are hard to remember and difficult to modify when we change the thermal simulation condition. PowerDC tool also accepts TCL (Tool Commands Language) commands in the batch mode run. We developed a TCL program to read the configure file as shown in Figure 4 and load input files specified in the configure file. The TCL code to implement the flow shown in Figure 5 is attached in Appendix III. This TCL code generates a run file of Cadence PowerDC commands to execute the following steps. If "run" is specified as 1 in the configure file, the run file will be submitted in PowerDC environment and display simulation results in PowerDC GUI environment. If "run" is specified as 0 in the configure file, only the run file is generated with no submission of the run file in PowerDC tool.

a) Reading three input files that are required to execute thermal simulation in PowerDC environment;
b) Executing thermal simulation;
c) Displaying simulation results or distribution of temperatures over the die in PowerDC GUI.


Process configuration file:
1. Read the configure file;
2. Assign parameters;
3. Check quality of parameters.

Process three input files:
1. Copy specified workspace file in configure file to local run directory and change link to local layout file. Update package material and structure parameters specified in configure file to the local workspace file;
2. Copy the specified layout file in configure file to local run directory and change link to specified power density file;
3. Check power density file. If is it a CSV or Excel file, this CSV file is converted to textual area-based power density file.

Prepare a TCL run file and execute run file in Cadence PowerDC:
1. Write to a run TCL file of Sigrity PowerDC commands for the automation of thermal analysis tasks;
2. Submit and execute the run file in Sigrity PowerDC environment if the "run" parameter is set to 1 in the configure file.

Figure 5: Flow to Generate Run File of PowerDC TCL Commands.

4 Power Density Files

The power density file models the power dissipation and distribution over the die that is required to run the thermal simulation. The accuracy of power density file is critical to the accuracy of the thermal simulation. Here are terminologies related to power dissipation and density calculation in the VLSI chip.

- *Static power:* power consumed while there is no circuit activities.
- *Dynamic power*: power consumed while inputs are active to charge and discharge circuits.
- *Average power*: the average value of power consumptions over the specified period.
- *Power density:* the average power consumption in the specified area of die.

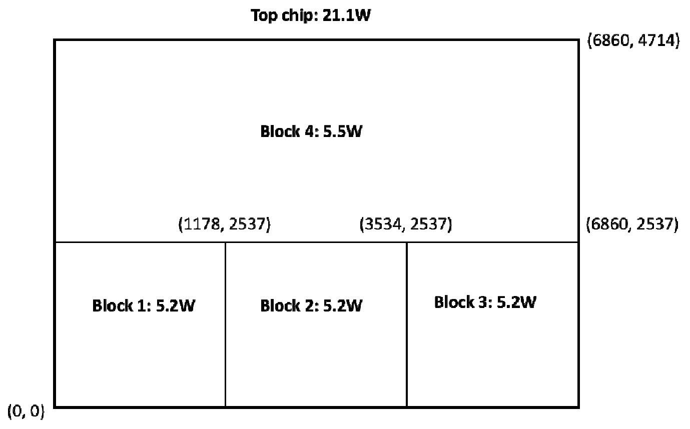
There are two formats are accepted by PowerDC tool to model the power dissipation and density of devices over the chip. The first format that is based on the average power numbers of macro blocks in the chip floorplan is used in the early design stage to choose the cooling technology and verify the temperature upper limit set by the design team for the circuit simulation. The second format is more accurate based on the even power grids over the chip layout to run the thermal simulation for the final signoff or tapeout of the design database to the manufacturing.

a) *Power density file in early design stage based on macro blocks in the floorplan*. The power density file is formed based on coordinates of top-level blocks in the floorplan and estimated power number for each macro block in the chip.
b) *Power density file based on evenly partitioned layout grids of chip layout*. The power density file contains the number of layout grids in the first line and then power numbers assigned to every layout grid area.

Figure 6 shows an example of power density file based on macro blocks in the chip floorplan. Figure 6(a) shows the floorplan than contains four top-level blocks in the chip and estimated power consumption numbers for each macro block. Figure 6(b) shows the power density file that corresponds to the floorplan of macro blocks and estimated power numbers shown in Figure 6(a). In details the power density file based on macro blocks has the following content:

a) The first line shows the name of top chip and (x y) coordinates of the top chip's bottom-left and top-right corners, and the estimated total power of the chip.
b) The second line as well as other lines show each macro block's name, (x y) coordinates of block's bottom-left and top-right corners, and estimated power consumption of this block.

The following methodology has been developed to construct the above power density file in early design stage. Designers use one spreadsheet table to fill in the information required to generate the power density file - coordinates of top-level blocks in the floorplan and estimated power consumptions of blocks. Then the design team can run a TCL function genPwrFile()shown in Appendix III to convert the information in the spreadsheet to the format of power density file in the text file. PowerDC tool will accept this power density file to simulate thermal effects on the chip in the early design stage.



(a) Floorplan and Power Estimation.

Figure 6a: Power Density File in Early Design Stage.

```
# Block name, X0 (um), Y0(um), X1 (um), Y1 (um), Power (W)
  Chip, 0, 0, 6860, 4714, 21.1
  Block1, 0, 0, 1178, 2537, 5.2
  Block2, 1178, 0, 3534, 2537, 5.2
  Block3, 3534, 0, 6860, 2537, 5.2
  Block4, 0, 2537, 6860, 4714, 5.5
```

(b) Power Density File

Figure 6b: Power Density File in Early Design Stage.

Many researches have been done how to estimate power consumption numbers based on different circuit types in the early design phase [27-29]. Table IV summarized methods and EDA tools we can use to estimate power consumptions for different circuit types. Figure 7 shows more details of power estimation flow for logic synthesized block. Figure 8 shows the flow to estimate power consumption for memory and analog circuits. Note that memory block can contain millions of devices hard to be simulated significantly. Fast Spice simulator and netlist reduction technique are necessary to simulate large-size memory circuit. One netlist reduction method based on the extracted netlist from SRAM memory layout was explained in [29].

Table IV. Power Estimation Methods and EDA Tools Based on Circuit Types.

TYPE	METHOD	EDA TOOL
DIGITAL LOGIC	SYNTHESIZE RTL CODE TO GATE NETLIST BASED ON CELLS LIBRARY AND TIMING CONSTRAITS. THE GATE NETLIST IS THEN SIMULATED BASED ON STIMULUS VECTORS TO ESTIMATE POWER CONSUMPTION.	LOGIC SYNTHESIS
ANALOG CIRCUIT	CIRCUIT IS SIMULATED BASED ON ESTIMATED INTERCONNECTS AND IMPUT VECTORS TO ESTIMATE POWER CONSUMPTION	CIRCUIT SIMULATOR
MEMORY CIRCUIT	USE FAST CIRCUIT SIMULATOR BASED ON ESTIMATED INTERCONNECTS AND REDUCED CIRCUIT NETLIST TO ESTIMATE POWER CONSUMPTION	CIRCUIT SIMULATOR

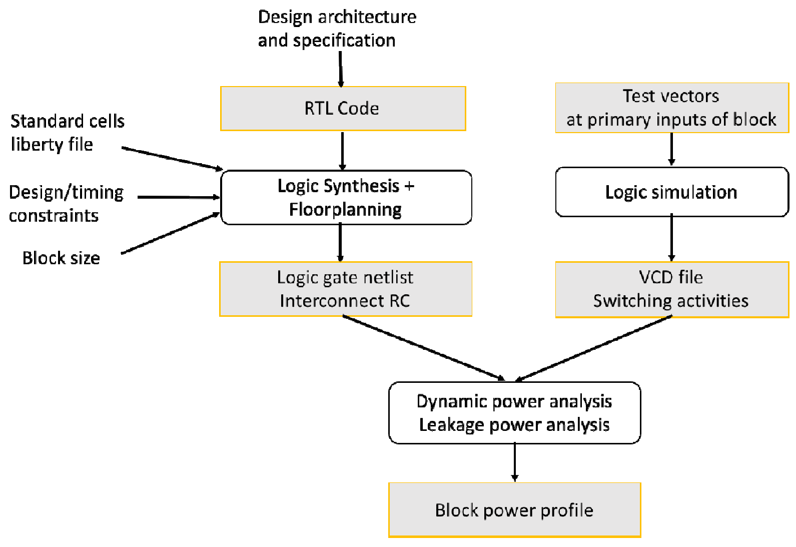

Design architecture and specification
RTL Code
Test vectors at primary inputs of block
Standard cells liberty file
Design/timing constraints
Block size
Logic Synthesis + Floorplanning
Logic simulation
Logic gate netlist Interconnect RC
VCD file Switching activities
Dynamic power analysis Leakage power analysis
Block power profile

Figure 7: Power Estimation Flow for Digital Logic Block.

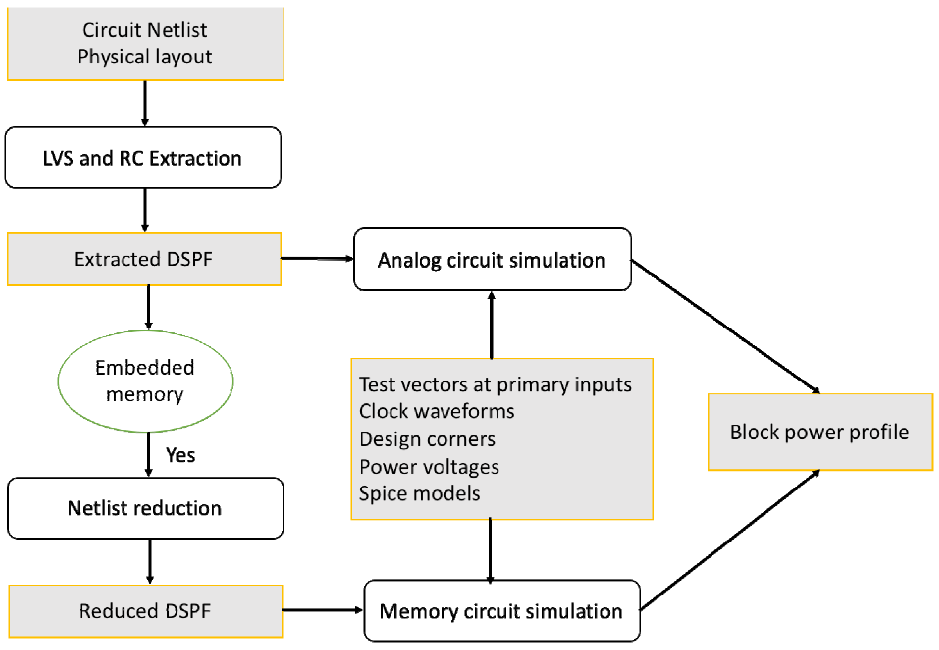


Figure 8: Power Estimation Flow for Memory and Analog Blocks.

The more accurate power density file used in the thermal signoff is shown in Figure 9. The chip layout is partitioned into even grids (i.e. 12 by 12 grids). The average power consumption number is assigned for each layout grid. Here is the theory behind the power density calculation. MOS devices are modelled as voltage-controlled current sources [22-24]. i(x,y,t) represents the current of devices at the location (x, y) and time (t). v(x,y,t) represents the supply voltage for this device at the location (x, y) and time (t). Note that the supply voltage v(x,y,t) are varied due to IR drop and di/dt noise on the power network. Power consumption is calculated as: *p(x,y,t) = i(x,y,t)*v(x,y,t)*. The average power consumption is the average of the above power equation over a specified period (T). The summation of average power consumptions for all devices in the specified layout grid area is defined as the power density that can be defined in the following equation:

$(\int \Sigma v(x_i, y_j, t) * i(x_i, y_j, t) dt)/T$ – including MOS devices located inside the specific area (1)

```
12   12
0.03 0.32 0.34 0.37 0.37 0.37 0.37 0.37 0.37 0.02 0.03 0.04
0.04 0.31 0.31 0.27 0.27 0.27 0.47 0.67 0.67 0.12 0.23 0.34
0.06 0.21 0.51 1.27 0.67 1.07 0.87 0.77 0.97 0.14 0.33 0.74
0.06 0.21 0.51 1.27 0.67 1.07 0.87 0.77 0.97 0.14 0.33 0.74
0.12 0.21 0.73 1.48 0.25 0.82 0.67 0.97 0.37 0.52 0.91 0.69
0.22 0.42 0.59 1.39 0.42 0.48 0.49 0.74 0.72 0.81 0.69 0.38
0.33 0.82 0.44 0.63 0.62 0.62 0.58 0.42 0.73 0.11 0.46 0.25
0.44 0.35 0.33 0.27 0.47 0.27 0.47 0.67 0.67 0.12 0.23 0.34
0.06 0.21 0.51 1.27 0.67 1.07 0.87 0.77 0.37 0.14 0.33 0.74
0.06 0.28 0.31 1.27 0.67 1.17 0.87 0.77 0.97 0.14 0.33 0.74
0.12 0.24 0.73 1.48 0.25 0.82 0.67 0.97 0.37 0.52 0.91 0.69
0.32 0.28 0.39 1.39 0.42 0.28 0.49 0.34 0.72 0.81 0.69 0.20
```

Figure 9: Layout Grid Based Power Density File for Thermal Signoff.

Figure 10 shows the overall flow to generate power density file based on partitioned layout grids. The flow starts from the circuit and layout database and extracts the layout netlist. Test vectors and VCD files from logic simulation are fed to power analysis tool in the dynamic simulation mode to get the power consumption profile of devices across the chip. In addition, we can adjust the size of layout grids (i.e. m x n) to improve the accuracy of thermal simulation result. A power network simulation tool Voltus from Cadence Design Systems was used to simulate the power consumption based on the extracted netlist from the full chip layout to get the power distribution profile [26]. The detailed setup and steps to simulate the power profile based on test vectors in the dynamic mode for power density file generation is shown in Figure 10. The worst-case or largest power numbers from the dynamic simulation are obtained and assigned to the power density file in order to get the worst-case temperature in the thermal simulation. Note that we would like to fix thermal violations based on the worst-case temperatures for the design success as well as the successful silicon in the manufacturing.

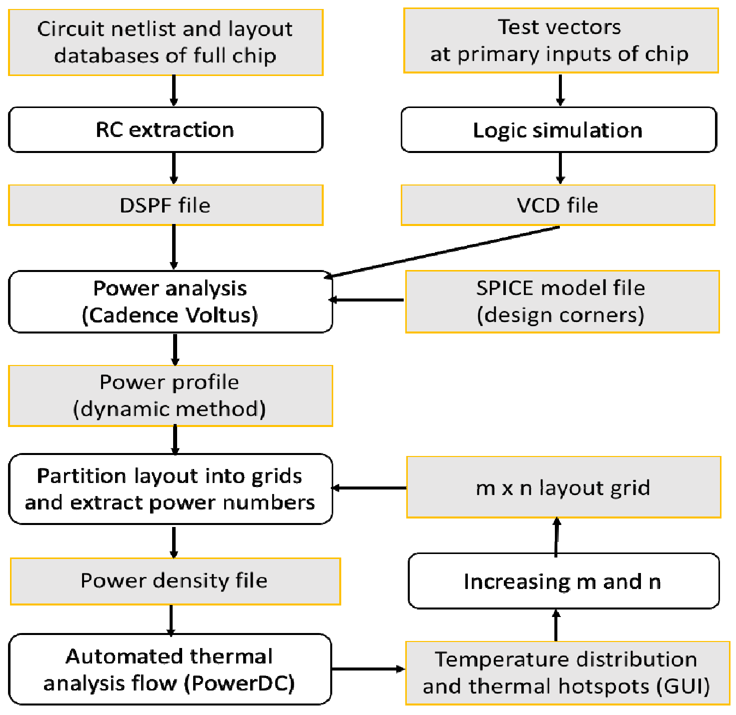
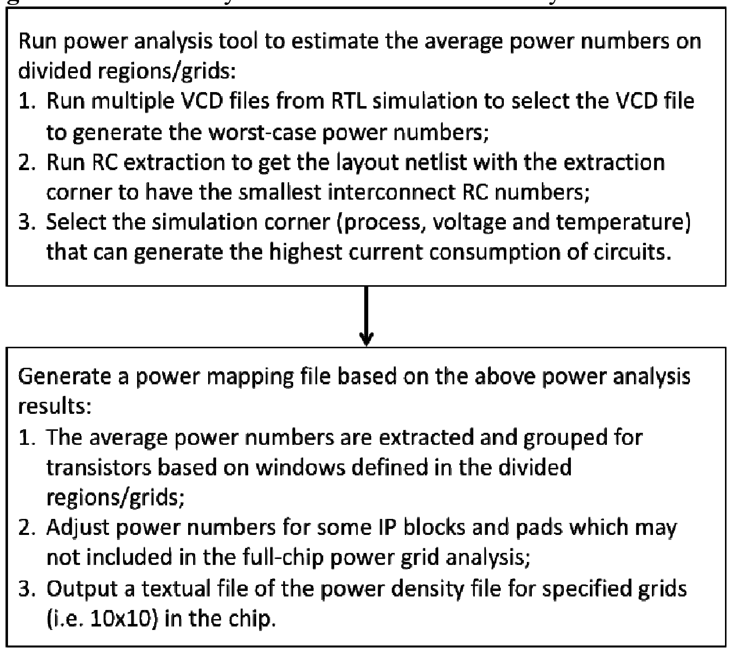

Circuit netlist and layout databases of full chip
Test vectors at primary inputs of chip
RC extraction
Logic simulation
DSPF file
VCD file
Power analysis (Cadence Voltus)
SPICE model file (design corners)
Power profile (dynamic method)
Partition layout into grids and extract power numbers
m x n layout grid
Power density file
Increasing m and n
Automated thermal analysis flow (PowerDC)
Temperature distribution and thermal hotspots (GUI)

Figure 10: Flow of Layout Grid-Based Power Density File Generation.


Run power analysis tool to estimate the average power numbers on divided regions/grids:

1. Run multiple VCD files from RTL simulation to select the VCD file to generate the worst-case power numbers;
2. Run RC extraction to get the layout netlist with the extraction corner to have the smallest interconnect RC numbers;
3. Select the simulation corner (process, voltage and temperature) that can generate the highest current consumption of circuits.

Generate a power mapping file based on the above power analysis results:

1. The average power numbers are extracted and grouped for transistors based on windows defined in the divided regions/grids;
2. Adjust power numbers for some IP blocks and pads which may not included in the full-chip power grid analysis;
3. Output a textual file of the power density file for specified grids (i.e. 10x10) in the chip.

Figure 11: Detailed Setup and Steps on On-chip Power Profile Simulation.

5 Experiment Results

The thermal simulation flow and thermal closure methodology represented in this paper has been adopted in one high-performance SOC design. Thermal simulation results in the early design stage help evaluate the cooling technology and verify on-chip temperature upper limit that is set for circuit design. In this SOC chip design project due to the high-speed requirement, design team sets a very aggressive upper temperature limit 80C in the circuit design. Choosing cooling technology for the semiconductor chip is important for the thermal management, reliability and cost. Heat sinks and liquid cold plates are two common cooling technologies used in VLSI systems [20]. Heat sinks cooling technology provides the low cost, but the efficiency to remove heats is low. Liquid cold plates technology is more expensive in the manufacturing but efficiency to remove heats is high. Liquid cold plates technology has been widely used in high-performance computer servers [21].

Table V shows thermal simulation results in this SOC design project at the early design phase based on heat sink cooling technology. It is reported in the [109.34C – 120.88C] temperature range over the die. The highest temperature 120.88C on the die violated the 80C upper bound limit set for the circuit design. Table VI shows temperature simulation results in cold plates technology that is reported in the [54.56C – 66.56C] temperature range. The highest temperature 66.56C in the die is below the 80C upper limit set for the circuit design. Cold plates cooling technology is selected for this SOC chip to meet the 80C upper temperature limit although cold plates cooling technology will be more expensive.

Table V. Thermal Simulation Results When Using Heat Sinks.

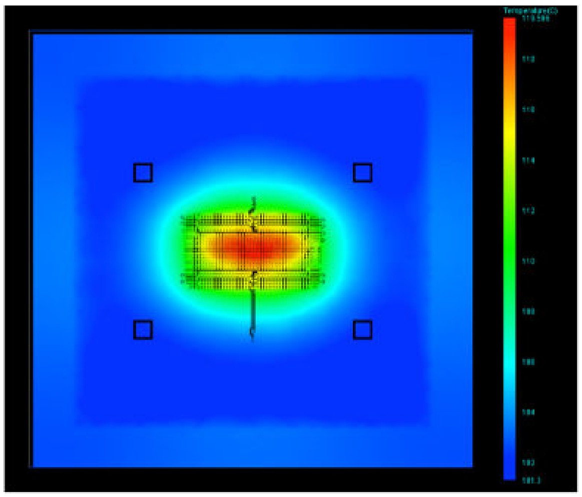
ITERATION	TOTAL POWER	AIR FLOW	AMBIENT TEMPERATURE	SIMULATED TEMPERATURES (DIE)
#1	12.23W	1.5M/S	55C	107.2C - 119.0C
#2	17.45W	1.5M/S	55C	113.2C – 125.6C
#3	15.90W	1.5M/S	55C	109.3C – 120.9C

Table VI. Thermal Simulation Results When Using Cold Plates.

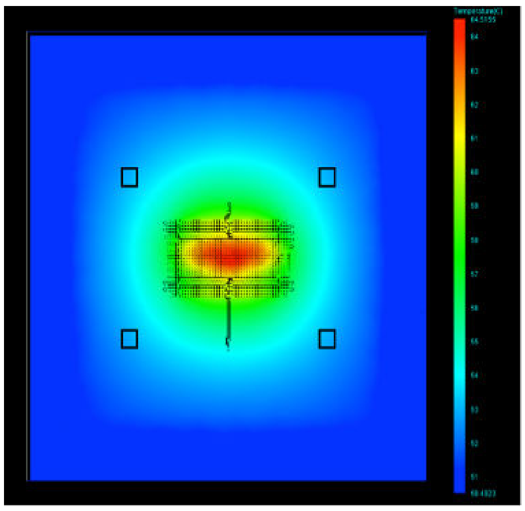
ITERATION	TOTAL POWER	AIR FLOW	AMBIENT TEMPERATURE	SIMULATED TEMPERATURES (DIE)
#1	12.23W	0M/S	55C	54.2C - 65.6C
#2	17.45W	0M/S	55C	54.8C – 66.8C
#3	15.90W	0M/S	55C	54.6C – 66.6C

Figure 12(a) and Figure 12(b) shows GUI plots of the thermal simulation result (design iteration #3 in Table V-VI). Designers can review the plot and find locations of the highest temperatures on the die. The red color in plots indicates the area of the highest temperatures in the die. In this design the highest temperatures are located in the middle of the die, since most of bus communications occur and hard to dissipate heats at the center of the die.

All the thermal simulation results in Table V-VI were executed and measured in Linux server RedHat 6.5OS and Intel Xeon® E5-2697 v4 @ 2.3GHZ platform. The total running time is less than 1 second for all test cases including steps of loading input files, building 3-D models, executing thermal simulation and displaying simulation plots.



(a) Heat Sinks



(b) Cold Plates

Figure 12: Plots of In-Die Temperature Distribution.

6 Conclusion

This paper describes thermal simulation flow and thermal closure methodology in one high-performance SOC design project. Thermal simulation flow adopted Cadence PowerDC tool as the core thermal simulation tool that is based on the thermal/electrical co-simulation for the accuracy. The tool provides rich GUI for designers to review and investigate thermal violations on the die. But the tool is difficult to setup and run the thermal simulation tool based on the GUI. Cadence PowerDC tool provides the commands interface to load input files and run the thermal simulation. We developed one automation flow based on an configure file and generate a run file of PowerDC commands in the sequence to load input files, executing thermal simulation and displaying temperature distribution results on GUI. The detailed of the configure file and power density files are described in this paper. We described formats of power density files and methods to construct power density files in the early design stage and tapeout. The methodology and flow presented in this paper can be applied to other design projects when thermal management becomes critical to the most of high-performance SOC chips.

Acknowledgment: Mike Kang, John Amann and CT Kao provided valuable inputs and comments on the flow development.

References:

1. Y. Jiao, H. Lin, P. Balaji and W. Feng, "Power and performance characterization of computational kernels on the GPU", Proc. of 2010 IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing, 2010.
2. S. Kalra, “Effect of temperature dependence on performance of Digital CMOS circuit technologies”, Proc. of International Conference on Signal Processing and Communication (ICSC), 2014.
3. R. Kumar, "Impact of temperature fluctuations on circuit characteristics in 180nm and 65nm CMOS technologies", Proc. of IEEE International Symposium on Circuits and Systems, 2006.
4. A . F. Robertson and D. Gross, “An electrical-analog method for transient heat-flow analysis”, Journal of Research of the National Bureau of Standards, Vol. 61, No. 2, August 1958.
5. M. Fawaz and F. N. Najm, “Fast simulation-based verification of RC power grids”, 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2016.
6. M. Tanaka, T. Kumamoto, S. Mori, “Alternative method for transient analysis of linear distributed RC networks”, IEEE Transactions on Circuits and Systems, Vol: 21, Issue: 6, Nov 1974.
7. "Sigrity PowerDC: DC and thermal analysis for packages and boards", https://www.cadence.com
8. “Cadence power integrity solutions for PCBs and IC packages”, https://www.flowcad.ch
9. “Industry-proven and foundry-certified analog and mixed-signal EM/IR solution”, https://www.ansys.com.
10. N. Chang, “Accurate thermal analysis, including thermal coupling of on-chip hot interconnect”, https://semiengineering.com.
11. “Introduction to FloTHERM”, https://www.mentor.com
12. W. Huang, "HotSpot-a chip and package compact thermal modeling methodology for VLSI design", Ph.D. Dissertation, University of Viginia, 2007.
13. T. J. Eguia, S. X.-D. Tan, D. Li, E. H. Pacheco, M. Tirumala and L. Wang, "General parameterized thermal modeling for high-performance microprocessor design", IEEE Transactions on Very Large Sscale Integration (VLSI) Systems, Vol. 20, No. 2, Feb 2012.
14. M. Pedram, S. Nazarian, "Thermal modeling, analysis and management in VLSI circuits: principles and methods", Proceedings of the IEEE, Vol 94, Issue 8, Aug. 2006.
15. H. Jung, P. Rong and M. Pedram, "Stochastic modeling of a thermally-managed multi-core system", Proc. of Design Automation Conference, 2008.

16. M. Garci, J.-B. Kammerer, L. Hebrard, "Towards electro-thermo-mechanical simulation of integrated circuits in standard CAD environment", Proc. of 20th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC), 2014.
17. Y.-K. Cheng, P. Raha, C.-C. Teng, E. Rosenbaum and Sung-Mo Kang, “ILLIADS-T: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol 17, Issue 8, Aug 1998.
18. Y.-H. Chang, C.-Y. Wang and Y.-A. Chen, "GA2CO: peak temperature estimation of VLSI circuits", Proc. of International SoC Design Conference (ISOCC), 2009.
19. T. J. Eguia, S. X.-D. Tan, R. Shen, D. Li, E. H. Pacheco, M. Tirumala and L. Wang, “General parameterized thermal modeling for high-performance microprocessor design”, IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, Vol 20, Issue 2, Feb. 2012.
20. R.C. Chu, R.E. Simons, M.J. Ellsworth, R.R. Schmidt and V. Cozzolino, “Review of cooling technologies for computer products”, IEEE Trans on Device and Materials Reliability, Vol 4, Issue 4, Dec. 2004.
21. M. J. Ellsworth, Jr., “Technical brief: design considerations for high performance processor liquid cooled cold plates”, https://www.electronics-cooling.com
22. M. Fawaz and F. N. Najm, "Fast simulation-based verification of RC power grids", 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2016.
23. M. Tanaka, T. Kumamoto, S. Mori, "Alternative method for transient analysis of linear distributed RC networks", IEEE Transactions on Circuits and Systems, Vol: 21, Issue: 6, Nov 1974.
24. D.-S. Cho, K.-H. Lee, G.-J. Jang, T.-S. Kim and J.-T. Kong, "Efficient modeling techniques for IR drop analysis in ASIC designs", Proc. Of Twelfth Annual IEEE International ASIC/SOC Conference, Sep 1999.
25. Q. K. Zhu, "Power distribution network design for VLSI", Wiley-Interscience, 2008.
26. “Voltus IC power integrity solution menu reference”, Cadence Design Systems Inc, 2019.
27. R. Eccles, “Making accurate power estimates at RTL”, https://semiengineering.com
28. N. Koduri and K. Vittal, "Power analysis of clock gating at RTL", https://www.design-reuse.com
29. C. Yin and M. Wang, "Spice circuit reduction for speeding up simulation and verification", Master thesis, Department of Electrical and Information Technology, Faculty of Engineering, Lund University, 2019.

Appendix I – Analogy Between Electrical Model and Thermal Model in VLSI System [4]

ELECTRI CAL MODEL	THERMAL MODEL
VOLTAGE (V)	TEMPERATURE (^oC)
CHARGE (C)	HEAT (J)
CURRENT (A)	HEAT FLOW (J/S)
RESISTANCE (OHM)	THERMAL RESISTANCE (K)
CAPACITANCE (F)	THERMAL CAPACITANCE (J/K)

Appendix II – Thermal/Thermal Co-Simulation Mechanism [8]

Thermal/Electrical co-simulation mechanism has been employed by PowerDC thermal simulation tool that is a commercial tool provided from Cadence Design Systems. It is known that the resistances of metal lines is related to the temperature based on the following equation: $R = R_0[1+\alpha(T-T_0)]$, where α is the temperature coefficient of resistance for metal material. Temperatures from thermal simulation can impact the resistance of metals and current modeling of devices. On the other side, resistance variance of metal lines in the power grid will impact voltage drops as well as the current consumptions of MOS devices. Therefore, thermal simulation needs to be integrated with IR drop analysis of the power grid and device current modelling in an iterative loop as shown in Figure 13.

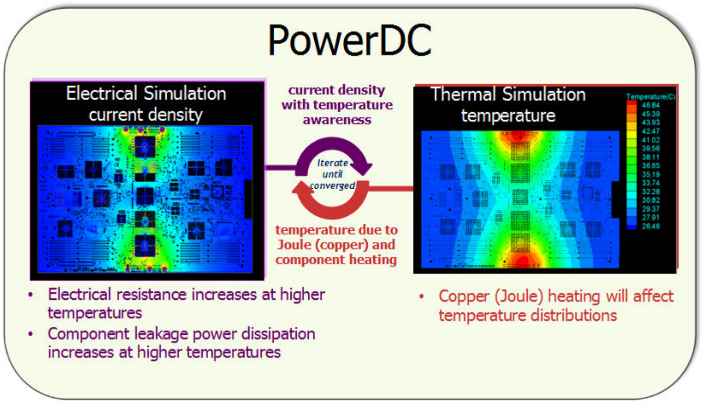

PowerDC
Electrical Simulation
current density
current density with temperature awareness
Thermal Simulation
temperature
Temperature(C)
46.84
45.39
43.93
42.47
41.02
39.56
38.11
36.65
35.19
33.74
32.28
30.82
29.37
27.91
26.46
Iterate until converged
temperature due to Joule (copper) and component heating
Electrical resistance increases at higher temperatures
Component leakage power dissipation increases at higher temperatures
Copper (Joule) heating will affect temperature distributions

Figure 13: Electrical/Thermal Co-Simulation Mechanism in Cadence PowerDC [8].

Appendix III – TCL Code of Thermal Simulation Flow.

```
#!/bin/env tclsh
#
# This is the TCL script to read a config file and run the PowerDC thermal simulation flow.
#
# The TCL procedure to generate power map file based on text file from Excel table of power numbers.
#
proc genPwrFile {csvFile} {
      set words [split $csvFile "/"]
      set last [lindex $words end]
      set pwrFile "$last.pwr.txt"
      set f1 [open $csvFile "r"]
      set lineNum 0
      while { [gets $f1 line] >= 0 } {
          incr lineNum
      }
      set blockNum [expr $lineNum - 2]
      close $f1
      set lineNum 0
      set f1 [open $csvFile "r"]
      # Print out the power map file
      set f2 [open $pwrFile "w"]
      while { [gets $f1 line] >= 0 } {
          incr lineNum
          if { $lineNum == 2 } {
              puts -nonewline $f2 "POWER_MAP_AREA"
              set words [split $line ","]
              for {set i 1} {$i < 5} {incr i} {
                  puts -nonewline $f2 " [lindex $words $i]"
              }
              puts -nonewline $f2 "\n"
```

```
            }
            if { $lineNum == 3 } {
                puts $f2 "AREA_POWER_MAP"
                puts $f2 $blockNum
            }
            if { $lineNum >= 3 } {
                set words [split $line ","]
                for {set i 1} {$i < 6} {incr i} {
                    if { $i ==1 } {
                        puts -nonewline $f2 "[lindex $words $i]"
                    } else {
                        puts -nonewline $f2 " [lindex $words $i]"
                    }
                }
                puts -nonewline $f2 "\n"
            }
        }
        close $f1
        close $f2
        return $pwrFile
}

#
# TCL procedure to get the index of words which is split from the line to include the key.
#
proc getWordsIndex {line key} {
        set words [split $line]
        set wlen [llength $words]
        set k -1
        for {set i 0} {$i < $wlen} {incr i} {
              if { [string match "$key=*" [lindex $words $i]] || [string match "$key*=*" [lindex $words $i]] }    {
                    set k $i
              }
        }
        return $k
}

#
# TCL procedure to replace the key of material properties in one line.
#
proc repMaterialField {line field newField} {
        set k [getWordsIndex $line $field]
        set words [split $line]
        if { $k > 0 } {
            set pat1 [lindex $words $k]
            set pat2 "$field=\"$newField\""
            regsub -all $pat1 $line $pat2 line2
        } else {
            puts "WARNING (repMaterialField procedure): $field cannot be found in line: $line"
            set line2 $line
        }
        return $line2
}

#
# Main program
#
if { $argc != 1 } {
      puts "Command: $argv0 <configFile>"
      exit
```

```
}
# Initialize key parameters
set curDir [pwd]
set wsfile ""
set layfile ""
set temperature ""
set component ""
set pwrfile ""
set csvfile ""
set tempfile "$curDir/tempMapFile.txt"

# If run powerDC simulation
set run 0

# Measure the elapse time of running
set startTime [clock seconds]
#
# Step 1: Read the config file
#
puts "Step 1: Read the config file"
set cfgFile [lindex $argv 0]
puts "Config file: $cfgFile"
if { ! [file isfile $cfgFile] } {
    puts stderr "No $cfgFile exists"
    exit 1
}
if {[catch {set f1 [open $cfgFile r]}]} {
    puts stderr "Could not open $cfgFile to read"
    exit 1
}
while { [gets $f1 line] >= 0 } {
    # if not comment line
    if { ! [string match "#*" $line] } {
        set line2 [join $line]
        set words [split $line2]
        set key [string tolower [lindex $words 0]]
        if [string equal $key "wsfile"] {
            set wsfile [lindex $words 1]
        } \
        elseif [string equal $key "layfile"] {
            set layfile [lindex $words 1]
        } \
        elseif [string equal $key "pwrfile"] {
            set pwrfile [lindex $words 1]
        } \
        elseif [string equal $key "csvfile"] {
            set csvfile [lindex $words 1]
        } \
        elseif [string equal $key "tempfile"] {
            set tempfile [lindex $words 1]
        } \
        elseif [string equal $key "run"] {
            set run [lindex $words 1]
        } \
        elseif [string equal $key "temperature"] {
            set temperature [lindex $words 1]
        } \
        elseif [string equal $key "airvelocity"] {
            set airvelocity [lindex $words 1]
        } \
        elseif [string equal $key "name"] {
            set name [lindex $words 1]
        } \
        elseif [string equal $key "type"] {
            set type [lindex $words 1]
        } \
        elseif [string equal $key "sinkmaterial"] {
            set sinkmaterial [lindex $words 1]
        } \
        elseif [string equal $key "length"] {
            set length [lindex $words 1]
        } \
        elseif [string equal $key "width"] {
            set width [lindex $words 1]
        } \
        elseif [string equal $key "basethickness"] {
            set basethickness [lindex $words 1]
        } \
        elseif [string equal $key "limbheight"] {
            set limbheight [lindex $words 1]
        } \
        elseif [string equal $key "limbthickness"] {
            set limbthickness [lindex $words 1]
        } \
        elseif [string equal $key "limbpitch"] {
            set limbpitch [lindex $words 1]
        } \
        elseif [string equal $key "fingerwidth"] {
            set fingerwidth [lindex $words 1]
        } \
        elseif [string equal $key "fingerperiod"] {
            set fingerperiod [lindex $words 1]
        } \
        elseif [string equal $key "diameter"] {
            set diameter [lindex $words 1]
        } \
        elseif [string equal $key "countl"] {
            set countl [lindex $words 1]
        } \
        elseif [string equal $key "countw"] {
            set countw [lindex $words 1]
        } \
        elseif [string equal $key "offsetl"] {
            set offsetl [lindex $words 1]
        } \
        elseif [string equal $key "offsetw"] {
            set offsetw [lindex $words 1]
        } \
        elseif [string equal $key "adhesivematerial"] {
            set adhesivematerial [lindex $words 1]
        } \
        elseif [string equal $key "adhesivethickness"] {
            set adhesivethickness [lindex $words 1]
        } \
        elseif [string equal $key "heattransfercoefficient"] {
            set heattransfercoefficient [lindex $words 1]
        } \
        elseif [string equal $key "basetemperature"] {
            set basetemperature [lindex $words 1]
        } \
        elseif [string equal $key "limbtemperature"] {
            set limbtemperature [lindex $words 1]
        } \
        elseif [string equal $key "ambientconditiontype"] {
            set ambientconditiontype [lindex $words 1]
        } \
        elseif [string equal $key "xlocation"] {
            set xlocation [lindex $words 1]
        } \
        elseif [string equal $key "ylocation"] {
            set ylocation [lindex $words 1]
        } \
    }
}
close $f1
puts "wsfile = $wsfile"
puts "layfile = $layfile"
puts "pwrfile = $pwrfile"
puts "csvfile = $csvfile"
puts "run = $run"
puts "temperature = $temperature"
puts "airvelocity = $airvelocity"
# Softlink input files
puts "Do the softlink for $wsfile $layfile $pwrfile"
if { [file isfile $wsfile] } {
    if [catch [exec /bin/ln -sf $wsfile]] {
        puts "WARN: failed to /bin/ln -sf $wsfile";
    }
}
if { [file isfile $layfile] } {
    if [catch [exec /bin/ln -sf $layfile]] {
        puts "WARN: failed to perform /bin/ln -sf $layfile";
    }
}
if { [file isfile $pwrfile] && $csvfile eq "" } {
    if [catch [exec /bin/ln -sf $pwrfile]] {
        puts "WARN: failed to perform /bin/ln -sf $pwrfile";
    }
}
if { [file isfile $csvfile] } {
    set pwrfile [genPwrFile $csvfile]
}
# Check the quality of parameters in the config file
if {[string equal $wsfile ""]} {
     puts "ERROR: No wsfile is specified in $cfgFile"
     exit 1
} else {
    if { ! [file isfile $wsfile] } {
         puts "ERROR: No $wsfile exists"
         exit 1
    }
}
if {[string equal $layfile ""]} {
     puts "ERROR: No layfile is specified in $cfgFile"
     exit 1
} else {
    if { ! [file isfile $layfile] } {
         puts "ERROR: No $layfile exists"
         exit 1
    }
```

```
}
if {[string equal $pwrfile ""] && [string equal $csvfile ""]} {
       puts "ERROR: No pwrfile or csvfile is specified in $cfgFile"
       exit 1
} else {
     if { ! [file isfile $pwrfile] && ! [file isfile $csvfile] } {
          puts "ERROR: No $pwrfile or $csvfile exists"
          exit 1
     }
}
if {[string equal $temperature ""]} {
       puts "ERROR: No temperature is specified in $cfgFile"
       exit 1
} else {
     if { [catch {expr {abs($temperature)}}] } {
          puts "ERROR: $temperature is not numerical value in $cfgFile"
          exit 1
     }
}
if {[string equal $airvelocity ""]} {
       puts "ERROR: No airvelocity is specified in $cfgFile"
       exit 1
} else {
     if { [catch {expr {abs($airvelocity)}}] } {
          puts "ERROR: $airvelocity is not numerical value in $cfgFile"
          exit 1
     }
}
#
# Step 2: copy three files (wsfile, layfile and pwrfile) and modified them in local run directory
#
# Local workspace and layout files in the run directory
set wsfile2 "$curDir/ws.local.pdcx"
set layfile2 "$curDir/lay.local.spd"
puts "Step 2: Modify the workspace spec file ($wsfile) to $wsfile2"
if {[catch {set wsf2 [open $wsfile r]}]} {
     puts stderr "Could not open $wsfile to read"
     exit 1
}
if {[catch {set wsf3 [open $wsfile2 w]}]} {
     puts stderr "Could not open $wsfile2 to write"
     exit 1
}
# Write to the local workspace file
while { [gets $wsf2 line] >= 0 } {
     if [string match "Layout=*" $line] {
          puts $wsf3 "Layout=\"$layfile2\""
     }
     elseif [string match "Temperature=*" $line] {
          puts $wsf3 "Temperature=\"$temperature\" AirVelocity=\"$airvelocity\" ThermalSetupVersion=\"1\""
     } \
     elseif [string match "*\<HeatSink Name*" $line] {
          # Package material specification
          # Remove heading white spaces of the line
          set line2    [string trim $line]
          regsub -all {\s+=} $line2 {=} line2su_424_0209_cold_plate_031017.pdcx
          set words [split $line2]
          # Replace the specified fields in package material properties
          if [info exists name] {
               set line2 [repMaterialField $line2 "Name" $name]
```

```
    }
    if [info exists type] {
        set line2 [repMaterialField $line2 "Type" $type]
    }
    if [info exists sinkmaterial] {
        set line2 [repMaterialField $line2 "SinkMaterial" $sinkmaterial]
    }
    if [info exists length] {
        set line2 [repMaterialField $line2 "Length" $length]
    }
    if [info exists width] {
        set line2 [repMaterialField $line2 "Width" $width]
    }
    if [info exists basethickness] {
        set line2 [repMaterialField $line2 "BaseThickness" $basethickness]
    }
    if [info exists limbheight] {
        set line2 [repMaterialField $line2 "LimbHeight" $limbheight]
    }
    if [info exists limbthickness] {
        # puts "DEBUG 1.0: limbthickness = $limbthickness"
        set line2 [repMaterialField $line2 "LimbThickness" $limbthickness]
    }
    if [info exists limbpitch] {
        set line2 [repMaterialField $line2 "LimbPitch" $limbpitch]
    }
    if [info exists fingerwidth] {
        set line2 [repMaterialField $line2 "FingerWidth" $fingerwidth]
    }
    if [info exists fingerperiod] {
        set line2 [repMaterialField $line2 "FingerPeriod" $fingerperiod]
    }
    if [info exists diameter] {
        set line2 [repMaterialField $line2 "Diameter" $diameter]
    }
    if [info exists countl] {
        set line2 [repMaterialField $line2 "CountL" $countl]
    }
    if [info exists countw] {
        set line2 [repMaterialField $line2 "CountW" $countw]
    }
    if [info exists offsetl] {
        set line2 [repMaterialField $line2 "OffsetL" $offsetl]
    }
    if [info exists offsetw] {
        set line2 [repMaterialField $line2 "OffsetW" $offsetw]
    }
    if [info exists adhesivematerial] {
        set line2 [repMaterialField $line2 "AdhesiveMaterial" $adhesivematerial]
    }
    if [info exists adhesivethickness] {
        set line2 [repMaterialField $line2 "AdhesiveThickness" $adhesivethickness]
    }
    if [info exists airvelocity] {
        set line2 [repMaterialField $line2 "AirVelocity" $airvelocity]
    }
    if [info exists heattransfercoefficient] {
        set line2 [repMaterialField $line2 "HeatTransferCoefficient" $heattransfercoefficient]
    }
    if [info exists basetemperature] {
```

```
                set line2 [repMaterialField $line2 "BaseTemperature" $basetemperature]
            }
            if [info exists limbtemperature] {
                set line2 [repMaterialField $line2 "LimbTemperature" $limbtemperature]
            }
            if [info exists ambientconditiontype] {
                set line2 [repMaterialField $line2 "AmbientConditionType" $ambientconditiontype]
            }
            if [info exists xlocation] {
                set line2 [repMaterialField $line2 "Xlocation" $xlocation]
            }
            if [info exists ylocation] {
                set line2 [repMaterialField $line2 "Ylocation" $ylocation]
            }
            puts $wsf3 "\t\t$line2"
        } else {
            puts $wsf3 "$line"
        }
}
close $wsf2
close $wsf3
puts "$wsfile2 is generated"
puts "Step 3: Modify the layout file ($layfile) to $layfile2"
if [catch {set layf2 [open $layfile r]}] {
    puts stderr "Could not open $layfile to read"
    exit 1
}
if [catch {set layf3 [open $layfile2 w]}] {
    puts stderr "Could not open $layfile2 to write"
    exit 1
}
# Write to the local layout file
while { [gets $layf2 line] >= 0 } {
    if [string match "*PowerDissipation =*" $line] {
        puts $layf3 "+                  PowerDissipation = Power, \"$pwrfile\""
    } elseif [string match "*OutputTempFileName =*" $line] {
        puts $layf3 "+                  OutputTempFileName = $tempfile"
    } else {
        puts $layf3 "$line"
    }
}
close $layf2
close $layf3
puts "$layfile2 is generated"
puts "Step 4: generate a powerdc_run.tcl file"
set tclfile "$curDir/powerdc_run.tcl"
if [catch {set tf [open $tclfile w]}] {
    puts stderr "Could not open $tf to write"
    exit 1
}
puts $tf "sigrity::open document \{$wsfile2\}"
if { ! [string equal $temperature ""] } {
     puts $tf "sigrity::update option -GlobalTemperature \{$temperature\}"
}
if { ! [string equal $airvelocity ""] } {
     puts $tf "sigrity::update option -ThermalAirVelocity \{$airvelocity\}"
}
puts $tf "sigrity::begin simulation"
puts $tf "puts \"Themal simulation is done\""
puts $tf "puts \"Temperature simulation results are saved in $tempfile \""
```

```
close $tf
puts "$tclfile is generated"
puts "Please review the $tclfile"
puts "The output temperature map file will be saved in $tempfile"
set endTime [clock seconds]
puts "Script running time: [expr $endTime - $startTime] seconds"
puts "\nYou can launch thermal analyis flow as follows: "
puts ">> powerdc -tcl $tclfile"
puts "      (or playTCL > select $tclfile)"
if { $run == 1 } {
     # Launch the powerDC thermal tool in GUI mode
     exec powerdc -tcl $tclfile
}
```