SpotLight: A Hotspot-Greedy, Light-Weighted, and Automated Thermal Modeling Framework for Early Smartphone Design

***Abstract*—This paper develops a hotspot-greedy, light-weighted, and automated framework, SpotLight, to achieve fast thermal analysis of smartphones at the pre-silicon stage. Besides, we apply superposed models and a designed least penalty problem to improve the accuracy. Experimental results show that the SpotLight can accurately simulate the system-level hotspot temperatures of smartphones with average runtime of 0.56 ms.**

I. Introduction

The tradeoff between performance and device temperature has become a significant issue for modern mobile devices [1]. Thus, many researchers focus on smartphones' power/ thermal modeling and energy-aware dynamic thermal management (DTM) at the post-silicon stage [1]–[6]. However, the pre-silicon thermal-aware analysis is more influential since the adjustment is minimal after the production of the chip and phone. Moreover, designers need efficient thermal simulation tools to survive time-to-market pressure, e.g., architectural system-on-chip (SoC) design features have to be determined three years ahead of time [7]. Besides, with the increasing power density, both skin (surface) and junction temperatures must be controlled for modern phones. High junction temperatures can hugely impact the lifespan of devices [1]. On the other hand, hot skin can hamper the user experience [4]. Thus, the system-level thermal model is needed.

The prior works relating to smartphones' pre-silicon and system-level thermal modeling methodologies are mainly limited to long simulation time. The computational fluid dynamics (CFD) tools, e.g., Ansys Icepak [8], take about one minute to simulate the full heatmap of smartphones for one time step. The finite difference method (FDM) based compact thermal models (CTM) [9]–[11] need about $10 \sim 60$ ms. However, with huge amount of test cases, chip and phone manufacturers require faster thermal analysis for thermal-aware performance prediction. Clearly, more effective thermal modeling methodologies are required for optimizing smartphone design.

We aim to develop a faster pre-silicon and system-level thermal modeling framework instead of full-heatmap simulators. Hence, we utilize a hotspot-greedy scheme to choose only potential hotspots to shorten the simulation. Fig. 1 illustrates the number of nodes comparison between the conventional model and the proposed hotspot-greedy scheme. Moreover, we ensure high accuracy by the principle of superposition and least penalty optimization. To our best knowledge, the proposed work is the first automatic pre-silicon thermal modeling framework targeting smartphones that can accurately simulate system-level hotspot transient temperatures under 1.0 ms. We name the proposed framework SpotLight. Our major contributions are:

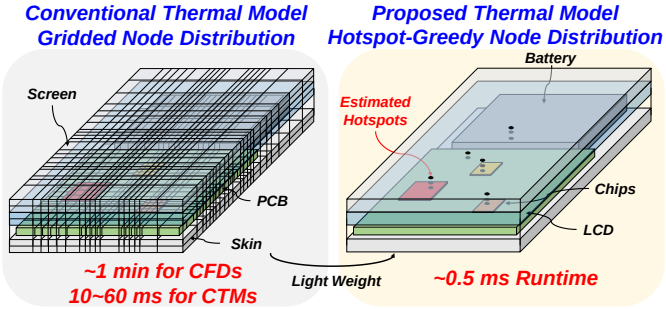


Fig. 1. Number of grid nodes and runtime comparison.

- We develop an automatic topology generator, Hotspot-AuToGen. It can automatically select the hotspots from the smartphone structure to generate the thermal Cauer resistor-capacitor (RC) topology [12]. Compared with full-heatmap techniques, our strategy of selecting only hotspots can significantly reduce the simulation runtime.
- We propose a thermal RC extractor, TRC-Extractor, to fit the temperature of each node. Different from the analytical method of heat transfer [13], we consider the thermal RC as variables and design a least penalty problem to find their values.
- We adopt the principle of superposition to improve the accuracy by increasing the number of RC parameters.

We organize this paper as follows. First, Section II introduces pre-silicon and system-level thermal modeling methodologies of prior arts, and states the problem formulation. Then, Section III details the proposed method, SpotLight. Finally, Section IV presents experimental results and Section V concludes this work.

II. Preliminaries and Problem Formulation

A. Artificial Method to Set Grids of CTM

Dousti *et al.* [9] and Lee *et al.* [10] utilized similar techniques to build system-level compact thermal simulators for providing the full-heatmap of a smartphone. The work in [9] artificially grids the components of smartphone. After building the thermal RC topology of the device, it calculates the thermal RC by the 3-D thermal resistance of conduction $r_{i,j} = r_{j,i} = r_i + r_j = (t_i/g_i + t_j/g_j)/A_{i,j}$ and lumped capacitance $C_{th,i} = \gamma \cdot c_i \cdot \rho_i \cdot V_i$. Here, t_i/t_j and g_i/g_j are the perpendicular distance and thermal conductivity of sub-component

i/j, respectively. $A_{i,j}$ is the area between two contacted sub-components. γ, c_i, ρ_i, and V_i are the correction factor, the specific heat, the density, and the volume of sub-component i, respectively. Moreover, it uses the pre-specified heat transfer coefficients (HTCs) and convection heat conductances to solve the boundary condition. The boundary thermal resistance is calculated as $r_{i,amb} = r_i + r_{amb} = (t_i/g_i + 1/h_{air})/A_{i,amb}$, where h_{air} is the HTC of air. The heat capacitance at boundary is calculated as $C^{bndry}_{th,i} = \gamma(c_i \cdot \rho_i \cdot V_i + C^{convn}_{th,\square} \cdot A_{i,amb})$, where $C^{convn}_{th,\square}$ is the convection heat capacitance.

According to the convergence validation in [9], it needs above $7,000$ sub-components to keep the error below 1% and its runtime of one time step is about 10 ms.

B. Adaptive Grid Builder

Lo *et al.* [11] developed a grid builder, DLAG-TA. It utilizes deep learning to automatically generate adaptive grids of handheld devices and can simulate full heatmaps of smartphones. The resolution level of generated grids is successfully parameterized: $Res^{x,y,z}_{b_j} = (\Delta T^{x,y,z}_{b_j}/level) \cdot n^{x,y,z}_{b_j}$, where $Res^{x,y,z}_{b_j}$, $\Delta T^{x,y,z}_{b_j}$, and $n^{x,y,z}_{b_j}$ are the resolution, maximum thermal gradient, and grids along the x, y, z-axis of block j. As long as users input the parameter *level*, DLAG-TA can automatically grid the components in the phone system. Thus, DLAG-TA can save users plenty of time to modulate the grid setting. As for calculating thermal RC, it uses the similar methodology as Therminator 2 [9].

According to the experimental results of [11], for different resolution levels and phones, DLAG-TA generates $22,000 \sim 52,000$ grids. The runtime of one time step is $13 \sim 56$ ms.

C. Problem Formulation

To speed up the simulation, we need to simplify the thermal model. Thus, we plan to retain only hotspots of critical places, including chips, printed circuit board (PCB), battery, display, and battery. To do this, it will encounter two problems.

Problem 1: Where are hotspots?

The hotspot tracking problem has been proven to be *NP-hard* [14]. Hence, in this work, we develop a heuristic Hotspot-AuToGen algorithm to predict the locations of hotspots, denoted as $node^{x,y,z}$.

Problem 2: Instead of using thousands of nodes such as in [9]–[11], how to use fewer nodes and also ensure high accuracy?

Different from the FDM-based works [9]–[11], we consider thermal resistors and capacitors as variables. Then, we adopt the principle of superposition to increase the number of variables, i.e., each heat source has its superposed model and RC parameters. Finally, we utilize the TRC-Extractor to extract these thermal RC variables and ensure the accuracy.

III. The SpotLight

Fig. 2 shows the proposed framework, SpotLight. First, given a smartphone's geometry and material information, we construct the CFD model and conduct the Hotspot-AuToGen algorithm to generate its thermal RC network topology. Next,

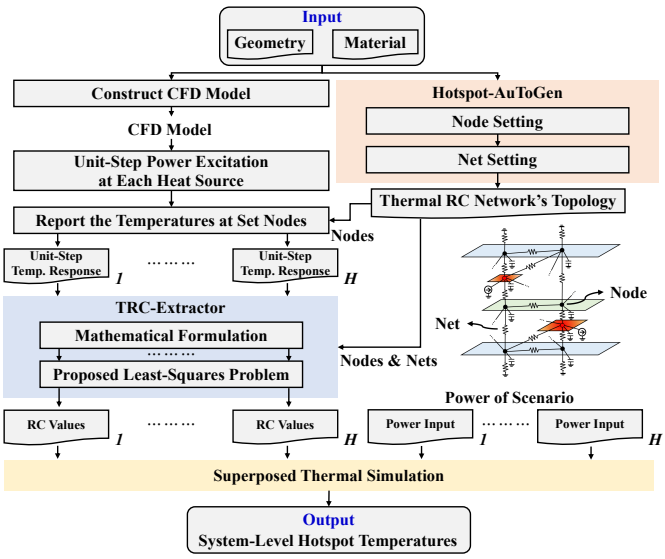


Fig. 2. The proposed framework, SpotLight.

we separately excite a unit-step power at each heat source and report their temperatures at those nodes generated by the Hotspot-AuToGen. Since there are H heat sources, we have H unit-step temperature response files. Then, in the TRC-Extractor, we sequentially execute mathematical formulation and our proposed least penalty problem to extract the RC values in each superposed model. Finally, given an operation scenario, we feed the power waveform of each heat source into its superposed model, complete each superposed model's thermal simulation, and superpose them.

A. Hotspot-AuToGen

Given a smartphone's geometry and material information, the Hotspot-AuToGen presented in **Algorithm 1** automatically generates the topology of hotspot-greedy thermal RC network.

Initially, we define the screen, PCB, and skin as the *specified layers* for follow-up node setting procedures. Next, we initialize a node vector $\mathbf{nv}_{hs}$ for storing hotspot nodes on heat sources. Then, we adopt a weighting way to determine hotspot nodes on heat sources.

We consider that the power density and the spatial distance between each two heat sources mainly determine the locations of hotspots. Hence, we sequentially calculate the power density weight vector $\mathbf{w}_p$ and distance weight matrix $\mathbf{D}$. Here, $\mathbf{w}_p$ is a $H \times 1$ vector, and $\mathbf{D}$ is a $H \times H$ matrix illustrating the distance between each pair of heat sources. For power density weights, we adopt the average power of several applications, e.g., games, daily of uses (DoUs), and benchmarks, and divide it by the area of the heat source. Then, we use the reciprocal of the distance between each pair of sources to obtain the distance weight matrix $\mathbf{D}$. Hence, the shorter the distance, the heavier the weight. We remark that $\mathbf{w}_p$ and $\mathbf{D}$ are user-specified. For instance, users can base $\mathbf{w}_p$ on the power of scenarios they're going to analyze.

Next, to predict the hotspot on a heat source, we calculate the impact of other heat sources according to the weights. (1)

Algorithm 1: Hotspot-AuToGen

Input: Geometry and material information
Set screen, PCB, and skin as the **specified layers**
Initialize the node vector $\mathbf{nv}_{hs}$
Calculate power density weight vector $\mathbf{w}_p$
Calculate distance weight matrix $\mathbf{D}$
for h_i = *1 to* H **do**
  Initialize x, y, and z to zeros
  Initialize h_i's total weight w_t to zero
  for h_j = *1 to* H **do**
    $x+ = f(\mathbf{w}_p(h_j), \mathbf{D}(h_i, h_j), x_c^{h_i}, x_c^{h_j}, x_l^{h_i}, x_l^{h_j})$
    $y+ = f(\mathbf{w}_p(h_j), \mathbf{D}(h_i, h_j), y_c^{h_i}, y_c^{h_j}, y_l^{h_i}, y_l^{h_j})$
    $z+ = f(\mathbf{w}_p(h_j), \mathbf{D}(h_i, h_j), z_c^{h_i}, z_c^{h_j}, z_l^{h_i}, z_l^{h_j})$
    $w_t+ = \mathbf{w}_p(h_j) \cdot \mathbf{D}(h_i, h_j)$
  end
  $[x\ y\ z]/ = w_t$
  Put $[x\ y\ z]$ into $\mathbf{nv}_{hs}$
  Set hotspots at $\mathbf{nv}_{hs}(h_i)$
end
for h_i = *1 to* H **do**
  (In the vertical direction of $\mathbf{nv}_{hs}(h_i)$)
  Set hotspots at the positions of **specified layers**
  for h_j = *1 to* H **do**
    if $h_i \neq h_j$ *and no* ***specified layers*** *between heat sources* h_i *and* h_j **then**
      Set hotspots at the places having the highest thermal conductivities between heat sources h_i and h_j
    end
  end
end
Perform **Detour Node Setting Procedure**
Perform **PCB Node Setting Procedure**
Fully connect all hotspots

can determine the relative position of heat sources h_i & h_j and calculate the impact of h_j on h_i. For instance, if the center point of h_j heat source is beyond the right boundary of h_i heat source, i.e., $x_c^{h_i} + x_l^{h_i}/2$, we will multiply the weight of h_j on h_i by the right boundary of h_i heat source. This approach can effectively consider their thermal coupling and ensure the obtained point is inside the h_i heat source.

$$f(\mathbf{w}_p(h_j), \mathbf{D}(h_i, h_j), \square_c^{h_i}, \square_c^{h_j}, \square_l^{h_i}, \square_l^{h_j})) = \begin{cases} \mathbf{w}_p(h_j)\mathbf{D}(h_i, h_j)(\square_c^{h_i} + \frac{\square_l^{h_i}}{2}), & \text{if } \square_c^{h_j} > \square_c^{h_i} + \frac{\square_l^{h_i}}{2} \\ \mathbf{w}_p(h_j)\mathbf{D}(h_i, h_j)(\square_c^{h_i} - \frac{\square_l^{h_i}}{2}), & \text{if } \square_c^{h_j} < \square_c^{h_i} - \frac{\square_l^{h_i}}{2} \\ \mathbf{w}_p(h_j)\mathbf{D}(h_i, h_j)\square_c^{h_j}, & \text{otherwise.} \end{cases} \quad (1)$$

Here, $\square_c$ is the center and $\square_l$ is the length in $\square$-axis. Then, for heat source h_i's hotspot, we will divide the sum of each source's impact (including the self-heating effect) by the total weight, w_t, and take it as the predicted coordinate. Finally, we will put the predicted node into node vector $\mathbf{nv}_{hs}$ and set a hotspot at $\mathbf{nv}_{hs}(h_i)$.

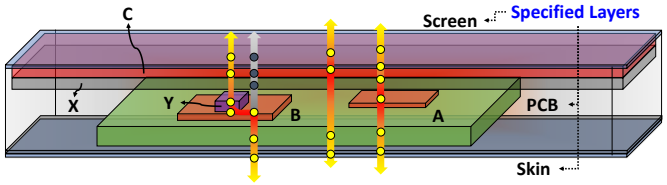


Fig. 3. The illustration of setting vertical and detour nodes. Red blocks are the heat soruces and yellow dots are the set hotspots by Hotspot-AuToGen. Gray dots are abandoned by Procedure 1 due to heat flow detour.

Procedure 1: Detour Node Setting

Input: Geometry and material information
Initialize the *detour_set*
for h_i = *1 to* H **do**
  (In the vertical direction of $\mathbf{nv}_{hs}(h_i)$)
  if *the low thermal conductivity block is met and there are blocks with high thermal conductivities nearby* **then**
    Assume that a heat flow detour happens here
    if *the block is not in detour_set* **then**
      Put the block into *detour_set*
    end
    Remove the block after the heat flow detour
  end
end
$N_D \leftarrow$ The size of *detour_set*
for i = *1 to* N_D **do**
  Set a hotspot at the center of *detour_set*(i)
  Set extra hotspots on the above and below layers of *detour_set*(i)
end

After that, we try to set nodes on the heat dissipation path for the precise fitting result. Since the structure of a smartphone is stack-like and its thickness along z-direction is small. Thus, heat flows are mainly vertical in the device [13]. As a result, we set hotspots in the vertical direction of $\mathbf{nv}_{hs}(h_i)$.

$$n_{h_i,l} = (\mathbf{nv}_{hs}(h_i)_x, \mathbf{nv}_{hs}(h_i)_y, z_l). \quad (2)$$

Here, $n_{h_i,l}$ indicates the set hotspot at l layer, which can be the specified layer or other blocks' z-plane. Thus, we first set hotspots at specified layers in the vertical direction of $\mathbf{nv}_{hs}(h_i)$. Then, we will set additional hotspots at the place having high thermal conductivity if there are no specified layers between a pair of heat sources. The goal is to obtain more accurate thermal coupling behavior between each pair of heat sources. Fig. 3 illustrates the process of setting vertical nodes. We take heat source A, which predicted hotspot is $\mathbf{nv}_{hs}(A)$, as an example. First, we set nodes at the corresponding positions of $\mathbf{nv}_{hs}(A)$ at specified layers, i.e., screen, PCB, and skin.

Procedure 2: PCB Node Setting

Input: Geometry and material information
for *heat source on PCB* **do**
  Set hotspots on the four sides of the heat source
end
Set the corresponding hotspots on the screen and skin

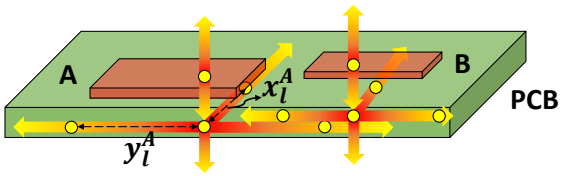


Fig. 4. The illustration of PCB node setting procedure.

Then, we set nodes at the corresponding positions of $\mathbf{nv}_{hs}(A)$, at other heat sources above or below it, i.e., source C. Next, we set one node at component X since there are no specified layers between A and C. The node setting process for other nodes in $\mathbf{nv}_{hs}$ is the same.

Then, we perform the **Detour Node Setting Procedure** presented in **Procedure 1** to track the path of the heat flow detour. We define the situation that the vertical dissipation path of an arbitrary $\mathbf{nv}_{hs}(h_i)$ contains low thermal conductivity material blocks such as air, and there are blocks with high thermal conductivity nearby them as the *heat flow detour*. We call those high thermal conductivity blocks as heat flow detour blocks. We declare a detour set, $detour\_set$, to store those heat flow detour blocks, and then, through the vertical path of each $\mathbf{nv}_{hs}(h_i)$, we check whether the condition of heat flow detour satisfies. When a heat flow detour happens, we insert the detour block into the $detour\_set$. After that, we set the center nodes of blocks in the $detour\_set$ as hotspot nodes. Meanwhile, to increase the model's accuracy, we add extra hotspot nodes located on the above and below layers of each heat flow detour block. The heat source B and block Y with high thermal conductivity in Fig. 3 can help readers understand this procedure.

Next, to deal with the horizontal heat flows on PCB, we perform the **PCB Node Setting Procedure** shown in **Procedure 2**.We set hotspot nodes around the four sides of each heat source on the PCB. Then, we also set their corresponding hotspot nodes on the screen and skin for modeling the dissipation paths to ambient. Fig. 4 illustrates this setting procedure.

Finally, we fully connect all hotspot nodes and complete the thermal RC topology as shown in Fig. 5.

B. TRC-Extractor

1) Mathematical Formulation: We develop the TRC-Extractor based on the duality between electrical and thermal circuits [13]. We utilize a simple thermal RC circuit shown in Fig. 6 to illustrate the mathematical formulation. There are two heat sources in the circuit shown in Fig. 6.(a). Since we adopt the principle of superposition as our modeling methodology,

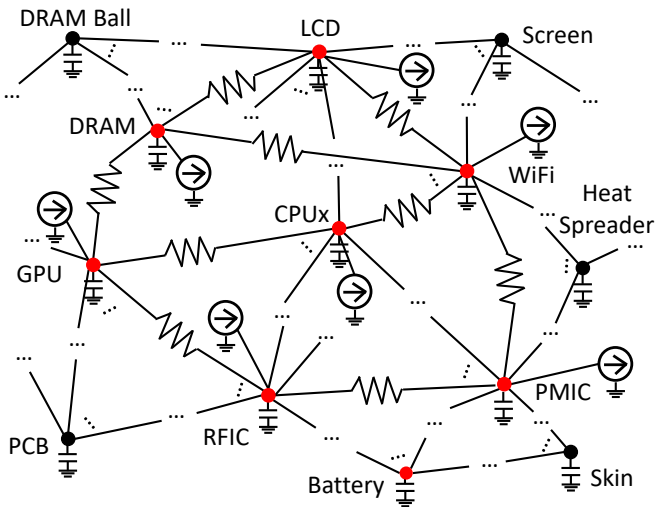


Fig. 5. The thermal RC network model. All nodes are fully-connected. Red nodes represent heat sources.

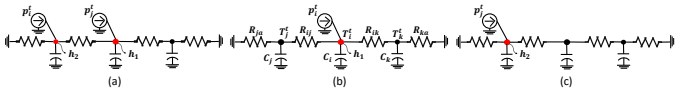


Fig. 6. (a) A circuit with two heat sources. (b) Its h_1-th superposed circuit. (c) Its h_2-th superposed circuit.

we split it to two superposed models as shown in Fig. 6.(b) and 6.(c). Both of them have the same topology but different thermal RC values. We will formulate the equations of h_1-th superposed circuit in Fig. 6.(b) to illustrate how we extract thermal RC values.

By the Kirchhoff's current law (KCL) and backward Euler difference method [15], the time-domain difference equations for the nodes are

$$\begin{aligned}
\frac{T_i^n - T_j^n}{R_{ij}} + \frac{T_i^n - T_k^n}{R_{ik}} + C_i\frac{T_i^n - T_i^{n-1}}{\Delta t} &= p_i^n,\\
\frac{T_j^n - T_i^n}{R_{ij}} + \frac{T_j^n - T_a}{R_{ja}} + C_j\frac{T_j^n - T_j^{n-1}}{\Delta t} &= 0,\\
\frac{T_k^n - T_i^n}{R_{ik}} + \frac{T_k^n - T_a}{R_{ka}} + C_k\frac{T_k^n - T_k^{n-1}}{\Delta t} &= 0.
\end{aligned} \tag{3}$$

Here, T_i^n is the temperature of node i at time $n\Delta t$, T_a is the ambient temperature, p_i^n is the power consumption at node i at time $n\Delta t$, and Δt is the time step size. R_{ij} is the thermal resistance between nodes i and j, and C_i is the thermal capacitance at node i.

As mentioned in Section III, we obtain the unit-step temperature responses as our training features by exciting a unit-step power to each heat source. Hence, the temperatures and power values in (3) are known and TRC-Extractor wants to determine values of thermal resistances and capacitances. We transfer the thermal resistance to thermal conductance $G = 1/R$, integrate

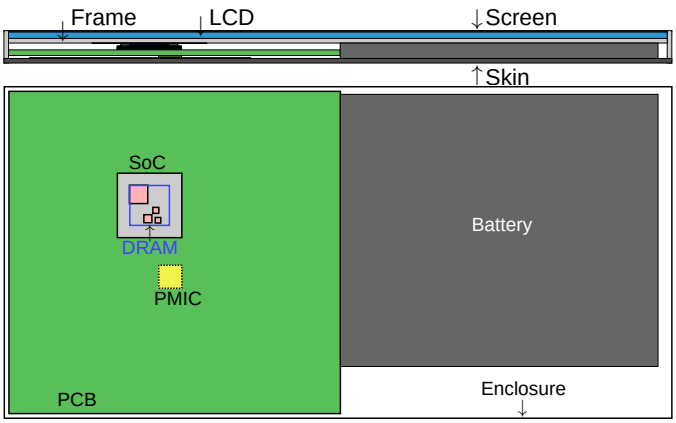


Fig. 7. Cross and longitudinal sections of the IDY-Phone CFD model. Sub-components are not shown.

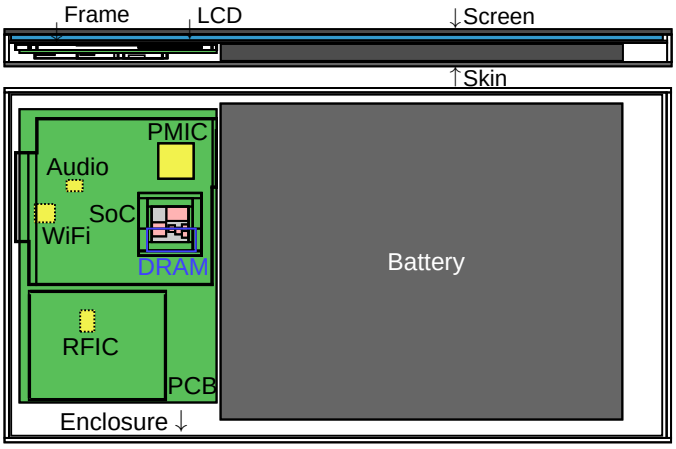


Fig. 8. Cross and longitudinal sections of CFD model for the OA55m. Sub-components are not shown.

time from Δt to $M\Delta t$, and rearrange (3) to be

$$\begin{bmatrix} T_{ij}^1 & T_{ik}^1 & 0 & 0 & \delta T_i^1 & 0 & 0 \\ T_{ji}^1 & 0 & T_{ja}^1 & 0 & 0 & \delta T_j^1 & 0 \\ 0 & T_{ki}^1 & 0 & T_{ka}^1 & 0 & 0 & \delta T_k^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ T_{ij}^n & T_{ik}^n & 0 & 0 & \delta T_i^n & 0 & 0 \\ T_{ji}^n & 0 & T_{ja}^n & 0 & 0 & \delta T_j^n & 0 \\ 0 & T_{ki}^n & 0 & T_{ka}^n & 0 & 0 & \delta T_k^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ T_{ij}^M & T_{ik}^M & 0 & 0 & \delta T_i^M & 0 & 0 \\ T_{ji}^M & 0 & T_{ja}^M & 0 & 0 & \delta T_j^M & 0 \\ 0 & T_{ki}^M & 0 & T_{ka}^M & 0 & 0 & \delta T_k^M \end{bmatrix}_{h_1} \begin{bmatrix} G_{ij} \\ G_{ik} \\ G_{ja} \\ G_{ka} \\ C_i \\ C_j \\ C_k \end{bmatrix}_{h_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix}_{h_1} . \quad (4)$$

Here, $T_{pq}^n = T_p^n - T_q^n$, $\delta T_p^n = (T_p^n - T_p^{n-1})/\Delta t$, and $T_{pa}^n = T_p^n - T_a$. We rewrite (4) for each heat source h as

$$\mathbf{A}_h \cdot \mathbf{x}_h = \mathbf{b}_h. \quad (5)$$

In addition, we use the similar procedure to formulate the steady-state equations as

$$\begin{bmatrix} T_{ij}^s & T_{ik}^s & 0 & 0 & 0 & 0 & 0 \\ T_{ji}^s & 0 & T_{ja}^s & 0 & 0 & 0 & 0 \\ 0 & T_{ki}^s & 0 & T_{ka}^s & 0 & 0 & 0 \end{bmatrix}_{h_1} \begin{bmatrix} G_{ij} \\ G_{ik} \\ G_{ja} \\ G_{ka} \\ C_i \\ C_j \\ C_k \end{bmatrix}_{h_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_{h_1} . \quad (6)$$

We rewrite (6) for each heat source h as

$$\mathbf{S}_h \cdot \mathbf{x}_h = \mathbf{d}_h. \quad (7)$$

Here, $\mathbf{x}_h$ needs to be determined.

2) Proposed least penalty problem: Since the variables of $\mathbf{x}_h$ in (5) and (7) are thermal conductances and capacitances, they should be positive. Additionally, (5) is over-determined. Moreover, since we utilize the principle of superposition to our modeling methodology, there will be H superposed thermal RC models if we have H heat sources.

Based on the above, we utilize the constrained least penalty problem to determine each $\mathbf{x}_h$ [16].

$$\begin{aligned} \min_{\mathbf{x}_h} \; & w_0 \cdot \|\mathbf{A}_h \cdot \mathbf{x}_h - \mathbf{b}_h\|_2^2 + w_1 \cdot \|\mathbf{S}_h \cdot \mathbf{x}_h - \mathbf{d}_h\|_2^2 \\ \text{s.t.} \quad & \mathbf{x}_h \geq \boldsymbol{\eta} \end{aligned} \quad (8)$$

Here, $\boldsymbol{\eta}$ is a positive user-specified vector.

We remark that we introduce the principle of superposition to our modeling methodology. In other words, each heat source has one set of RC parameters under the same topology. The reason is that we want to improve the accuracy by adding variables, i.e., the RC variables of superposed models. Thus, after the topology is decided, we excite each heat source with a unit-step power to obtain its unit-step temperature response. A well-chosen set of thermal RC variables can accurately fit all nodes' unit-step temperature response waveforms in each superposed model.

TABLE I. The specifications of tested smartphones.

Smartphone	Size (mm)	#Grids	#Sources
IDY	94.5×195.5×10.1	1,598,877	8
OA55m	80.0×150.5×8.4	978,908	13

C. Superposed Thermal Simulation

We conduct the transient thermal simulation by the modified nodal analysis (MNA) [15]. For the h-th superposed model, its temperatures $\mathbf{T}_h^n$ can be solved by

$$\mathbf{G}_h \mathbf{T}_h^n + \mathbf{C}_h \frac{\mathbf{T}_h^n - \mathbf{T}_h^{n-1}}{\Delta t} = \mathbf{p}_h^n, \quad (9)$$

where $\mathbf{G}_h$ and $\mathbf{C}_h$ are the thermal conductance and capacitance matrices, respectively, and $\mathbf{p}_h^n$ is the power vector.

Finally, we can get the temperatures of multiple heat sources $\mathbf{T}^n$ at each time $n\Delta t$ by summing all superposed models' temperatures as

$$\mathbf{T}^n = \sum_h \mathbf{T}_h^n. \quad (10)$$

TABLE II. Temperature error, distance error, and speed comparison between Ansys Icepak [8] and SpotLight.

Case	Total Time (s)	Avg. Pbat (W)	Hotspot-AuToGen Nodes Temperature RMSE (°C)	Hotspot-AuToGen Nodes Temperature MaxE (°C)	Hotspot-AuToGen Nodes Temperature MAPE (%)	Screen/ SoC/ PCB/ Skin Hotspots Temperature RMSE (°C)	Screen/ SoC/ PCB/ Skin Hotspots Temperature MaxE (°C)	Screen/ SoC/ PCB/ Skin Hotspots Temperature MAPE (%)	Screen/ SoC/ PCB/ Skin Hotspots Distance RMSE (mm)	Screen/ SoC/ PCB/ Skin Hotspots Distance MaxE (mm)	Screen/ SoC/ PCB/ Skin Hotspots Distance MAPE (%)	Ansys Icepak Time (s)	SpotLight Time (s)
IDY-C1	300	4.41	0.13	0.36	1.23	0.18	0.55	1.99	0.35	2.32	1.16	28801.28	0.09004
IDY-C2	300	6.81	0.25	0.80	1.33	0.37	1.37	2.13	0.43	2.33	1.13	28854.54	0.08977
IDY-C3	300	6.46	0.18	0.77	1.21	0.29	1.24	1.86	0.23	2.40	1.21	28846.12	0.09234
IDY-R1	100	0.35	0.06	0.16	1.30	0.08	0.34	2.03	0.19	2.30	1.24	3183.77	0.06505
IDY-R2	100	0.70	0.08	0.33	1.36	0.11	0.44	2.15	0.24	2.10	1.34	3122.19	0.06713
IDY-R3	100	1.85	0.12	0.37	1.50	0.18	0.70	2.30	0.36	2.21	1.40	3011.64	0.06629
IDY-B1	36	5.30	0.11	0.45	1.54	0.16	0.70	2.38	0.49	2.77	1.52	3466.32	0.01070
OA55m-C1	100	1.46	0.16	0.61	1.25	0.25	0.98	2.04	0.19	2.13	1.18	2760.55	0.06817
OA55m-C2	100	3.43	0.20	0.83	1.36	0.28	1.34	1.87	0.19	1.95	1.20	2760.30	0.06836
OA55m-C3	100	4.23	0.22	0.95	1.20	0.36	1.46	2.21	0.17	2.25	1.11	2760.44	0.06680
OA55m-R1	100	0.60	0.08	0.26	1.68	0.13	0.38	2.46	0.26	2.66	1.37	3183.77	0.06718
OA55m-R2	100	1.20	0.12	0.38	1.46	0.18	0.58	2.25	0.33	2.59	1.40	3122.19	0.06665
OA55m-R3	100	1.80	0.15	0.48	1.65	0.22	0.77	2.51	0.41	2.76	1.47	3011.64	0.06602
OA55m-B1	651	2.56	0.27	1.16	1.58	0.41	1.91	2.56	0.42	2.66	1.54	25808.92	0.43693
Avg.					1.40			2.19			1.30		

IV. Experimental Results

We implement SpotLight in C++ and MATLAB languages. We execute SpotLight and Ansys Icepak simulation on the machine with AMD Ryzen Threadripper 2950X 16-core processor, 3.50 GHz CPU, and 96 GB memory. The Ansys Icepak simulation is run in parallel with 16 processors.

A. The Generalization of Automation

We validate the generalization of automation by testing it on two smartphones. They are an industry smartphone (IDY) and the modified OPPO A55 [17] (OA55m). Table I lists their specifications, including size, number of grids in Ansys Icepak, and number of heat sources. Fig. 7 and Fig. 8 show the geometry of the CFD model for the IDY and OA55m. For IDY, the Hotspot-AuToGen automatically generates a thermal RC network with 98 nodes, $4,851$ thermal resistors, and 98 thermal capacitors. And for OA55m, there are 141 nodes, $10,011$ thermal resistors, and 141 thermal capacitors in the generated thermal RC network.

B. The Accuracy of SpotLight

To verify the accuracy of SpotLight, we generate several power patterns. Cases IDY-C1, IDY-C2, IDY-C3, OA55m-C1, OA55m-C2, and OA55m-C3 are the constant workload patterns with the power of selected applications, e.g., games and benchmarks. Then, we generate six normal distributed power cases for IDY and OA55m. The power of each heat source is $N(0.05, 0.016)$ W for IDY-R1 and OA55m-R1. IDY-R2 and OA55m-R2 have $N(0.1, 0.03)$ W for each heat source. And, IDY-R3 and OA55m-R3 have $N(0.15, 0.05)$ for each heat source. Case IDY-B1 is the benchmark power pattern adopting the power of benchmark Geekbench 6 [18]. Finally, OA55m-B1 uses the power of Antutu benchmark [19]. To describe the characteristics of these cases, we show their total time and average power of battery (Pbat) in Table II.

We use the root mean square error (*RMSE*), maximum error (*MaxE*), and mean absolute percentage error (*MAPE*) to evaluate our simulation accuracy. Table II shows the results, and we use two metrics to quantify the accuracy of SpotLight.

Firstly, to validate the fitting accuracy, we calculate temperature errors of the nodes generated by Hotspot-AuToGen. The MaxE is less than 1.16 °C, the RMSE is less than 0.27 °C, and the average MAPE is 1.40%. Fig. 9 shows the temperature comparison for the case OA55m-B1. We can see that the results of SpotLight successfully fit the ground truth, Ansys Icepak.

Secondly, we calculate temperature and distance errors of hotspots on screen, SoC, PCB, and skin. The result shows that the hotspot temperatures predicted by SpotLight are close to Icepak. The MaxE is less than 1.91 °C, the RMSE is less than 0.41 °C, and the average MAPE is 2.19%. And the spatial distance of the hotspots predicted by SpotLight and Icepak is close, too. The MaxE is less than 2.77 mm, and the RMSE is less than 0.49 mm. We divide the distance difference by the diagonal length of the screen, SoC, PCB, and skin for calculating the distance MAPE, and the average MAPE is only 1.30%. We can also observe the locations of hotspots in Fig. 10 and Fig. 11. Although at different times, the temperature nodes set by SpotLight are close to the hotspots simulated by Icepak.

C. The Efficiency of SpotLight

Table II lists the runtime of Ansys Icepak and SpotLight. The average runtime of Ansys Icepak is 54.57 s for one time step. In contrast, the average runtime of SpotLight is 0.00056 s for one time step. It shows that SpotLight can efficiently simulate system-level hotspot temperatures for early smartphone designs. Besides, due to fewer nodes, it is clear that SpotLight is also faster than full-heatmap CTMs [9]–[11]. Hence, analyzers can take advantage of SpotLight's high efficiency to simulate the cases with a high sampling rate. We take Antutu benchmark [19] for an instance. The

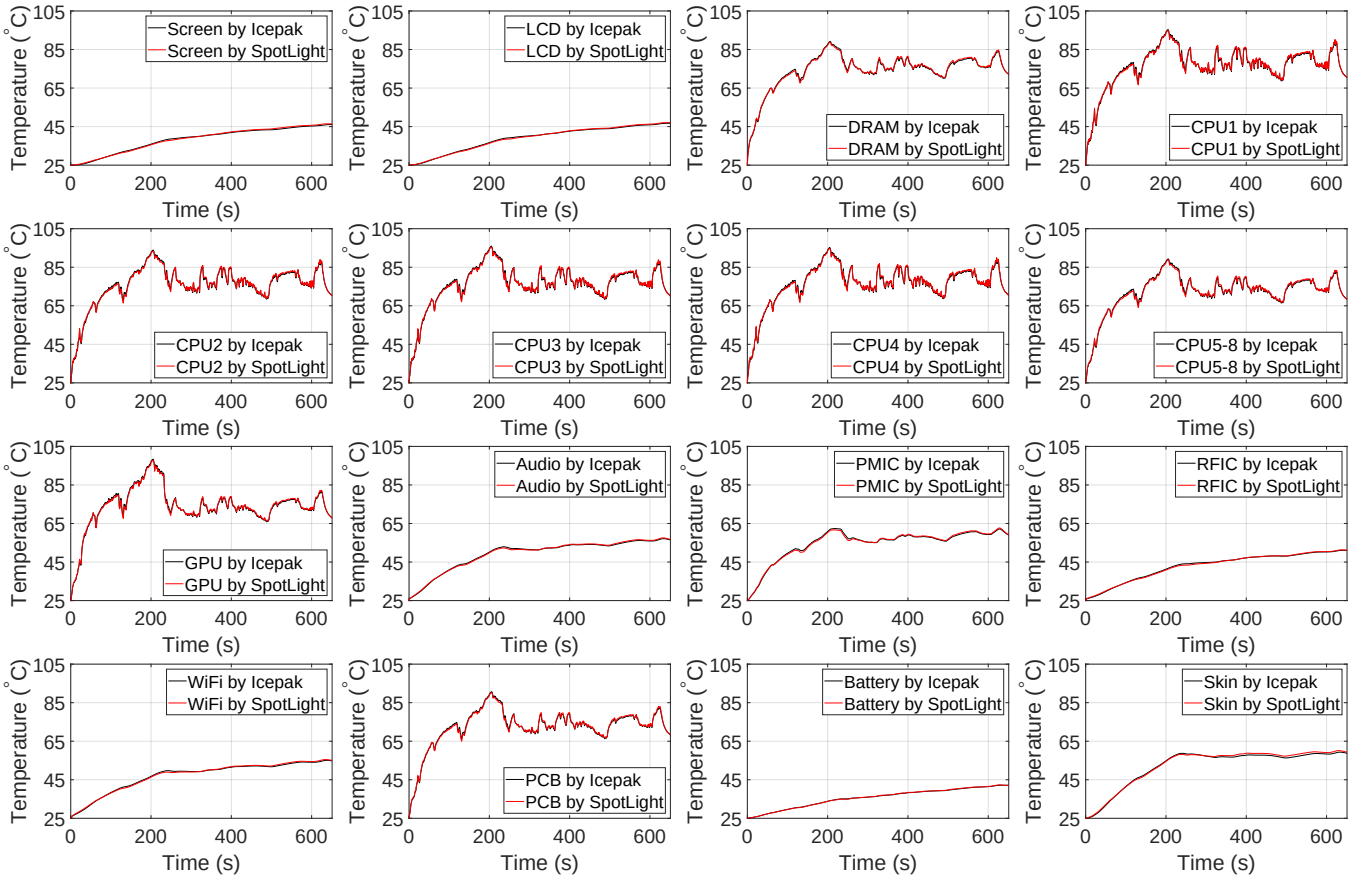
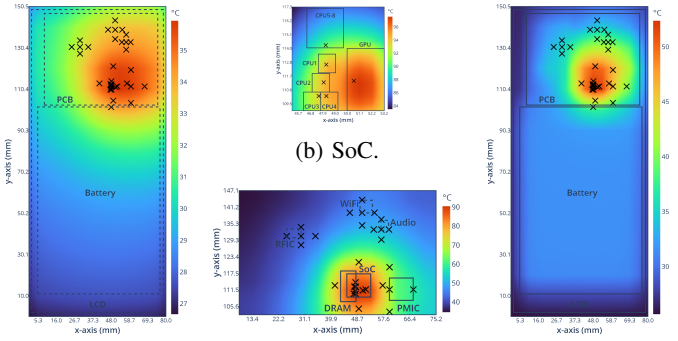
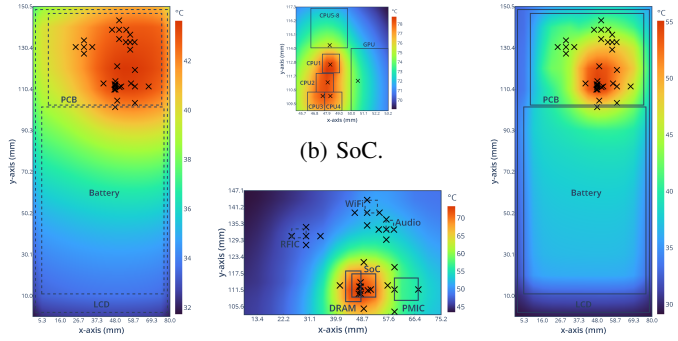


Fig. 9. The temperature comparison for case OA55m-B1.



(a) Screen. (b) SoC. (c) PCB. (d) Skin.

Fig. 10. Thermal maps run by Ansys Icepak and temperature nodes generated by SpotLight. (OA55m-B1 at 200 s)



(a) Screen. (b) SoC. (c) PCB. (d) Skin.

Fig. 11. Thermal maps run by Ansys Icepak and temperature nodes generated by SpotLight. (OA55m-B1 at 500 s)

smartphone's dynamic voltage and frequency scaling (DVFS) controller adjusts the frequency and voltage about every 10 ms. Hence, a 651 s Antutu scenario will have 65,100 time steps. The CFD tools, e.g., Ansys Icepak, will take over one month to simulate the pattern. However, SpotLight only needs 35 s. Moreover, chip manufacturers must analyze chip power across process, voltage, and temperature (PVT) space [20]. Hence, for each benchmark, designers must conduct thousands of thermal simulations due to enormous permutations of corners. SpotLight can help designers complete this challenging task in a short time.

D. The Influence of Detour and PCB Node Setting

To inspect the influence of detour and PCB node setting, we built different models *with*/*without* executing Procedure 1 and 2. Fig. 12 shows the comparison. It is clear that error increases while building the model without applying detour or PCB node setting. Among them, the PCB node setting is particularly influential. The RMSE is larger than 3.96 °C over four cases while not applying the PCB node setting. The reason is that

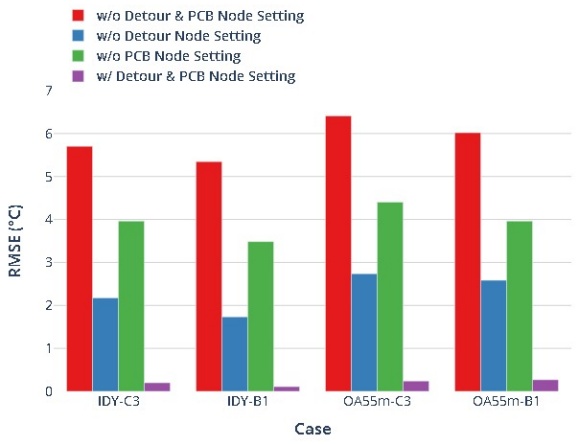


Fig. 12. The influence of detour and PCB node setting.

PCB is a primary horizontal heat dissipation path. There are many heat sources on it. Setting nodes around heat sources on PCB can improve the model's accuracy. Secondly, the detour node setting is also essential. The RMSE reaches 2.74 °C over four cases while not applying the detour node setting. The reason is that the heat dissipation path changes the direction at detours. Hence, we must track the dissipation path by the detour node setting procedure.

V. Conclusion

We have developed a hotspot-greedy, light-weighted, and automated thermal modeling framework, SpotLight, for early smartphone designs. We effectively narrow the region of potential hotspots and save the runtime of the simulation. The SpotLight can help pre-silicon and system-level thermal analysis be more efficient.

References

[1] B. Ozceylan, B. R. Haverkort, M. de Graaf, and M. E. T. Gerards, "Minimizing the maximum processor temperature by temperature-aware scheduling of real-time task," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 8, pp. 1084–1097, 2022.

[2] E. W. Wächter, C. de Bellefroid, K. R. Basireddy, A. K. Singh, B. M. Al-Hashimi, and G. Merrett, "Predictive thermal management for energy-efficient execution of concurrent applications on heterogeneous multicores," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 6, pp. 1404–1415, 2019.

[3] G. Bhat, G. Singla, A. K. Unver, and U. Y. Ogras, "Algorithmic optimization of thermal and power management for heterogeneous mobile platforms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 544–557, 2018.

[4] B. Egilmez, G. Memik, S. Ogrenci-Memik, and O. Ergin, "User-specific skin temperature-aware DVFS for smartphones," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 1217–1220, 2015.

[5] S. Kang, H. Choi, S. Park, C. Park, J. Lee, U. Lee, and S.-J. Lee, "Fire in your hands: Understanding thermal behavior of smartphones," in *Annual International Conference on Mobile Computing and Networking*, pp. 544–557, 2019.

[6] J. Park, S. Lee, and H. Cha, "Accurate prediction of smartphones' skin temperature by considering exothermic components," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 1500–1503, 2018.

[7] M. Hill and V. J. Reddi, "Gables: A roofline model for mobile SoCs," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 317–330, 2019.

[8] Ansys Icepak 19.0, https://www.ansys.com/products/electronics/ansys-icepak.

[9] M. J. Dousti, Q. Xie, M. Nazemi, and M. Pedram, "Therminator 2: A fast thermal simulator for portable devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 12, pp. 2528–2541, 2021.

[10] Y. M. Lee, H. W. Chiou, S. Y. Shiau, C. W. Pan, and S. H. Ting, "Phone-nomenon 2.0: A compact thermal model for smartphones," *IET Computers & Digital Techniques*.

[11] W. S. Lo, H. W. Chiou, S. C. Hsu, and Y. M. Lee, "DLAG-TA: Deep learning-based adaptive grid builder for system-level thermal analysis," in *IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, pp. 477–483, 2021.

[12] E. Cauer, W. Mathis, and R. Pauli, "Life and work of Wilhelm Cauer (1900–1945)," in *Proceedings of International Symposium on Mathematical Theory of Networks and Systems*, 2000.

[13] F. P. Incropera, D. P. DeWitt, T. L. Bergman, and A. S. Lavine, *Principles of Heat and Mass Transfer, global edition*. John Wiley & Sons, 2013.

[14] S. Reda, R. Cochran, and A. N. Nowroz, "Improved thermal tracking for processors using hard and soft sensor allocation techniques," *IEEE Transactions on Computers*, vol. 60, no. 6, pp. 841–851, 2011.

[15] F. N. Najm, *Circuit Simulation*. John Wiley & Sons, 2010.

[16] S. Boyd and L. Vandenberghem, *Convex Optimization*. Cambridge University Press, 2004.

[17] OPPO A55 Specifications, https://www.gsmarena.com/oppo_a55-11114.php.

[18] Geekbench 6 benchmark, https://www.geekbench.com/.

[19] Antutu benchmark, https://www.antutu.com/en/index.htm.

[20] S. Lichtensteiger and J. Bickford, "Using selective voltage binning to maximize yield," *IEEE Transactions on Semiconductor Manufacturing*, vol. 26, no. 4, pp. 436–441, 2013.