HotGauge: A Methodology for Characterizing Advanced Hotspots in Modern and Next Generation Processors

Alexander Hankin*§, David Werner*§, Maziar Amiraski*, Julien Sebot†, Kaushik Vaidyanathan‡, Mark Hempstead*

*Tufts University, Medford, MA, USA
†Intel Corp., Hillsboro, OR, USA
‡Google Inc., Sunnyvale, CA, USA
{alexander.hankin, david.werner}@tufts.edu

***Abstract*—On-chip thermal hotspots are becoming one of the primary design concerns for next generation processors. Industry chip design trends coupled with post-Dennard power density scaling has led to a stark increase in localized and application-dependent hotspots. These "advanced" hotspots cause a variety of adverse effects if untreated, ranging from dramatic performance loss, incorrect circuit operation, and reduced device lifespan. In the past, hotspots could be addressed with physical cooling systems and EDA tools; however, the severity of advanced hotspots is prohibitively high for conventional thermal regulation techniques alone. Fine-grained, architecture-level techniques are needed. To develop these new techniques, the architecture community needs the methods and metrics for simulating and characterizing advanced hotspots.**

This work presents a novel hotspot characterization methodology for modern and next generation processors which we have coined, HotGauge. HotGauge includes new methods and metrics to enable architects to build hotspot mitigation techniques of the future. To demonstrate the utility of HotGauge, we present a case study in which we characterize the hotspot behavior in a modern 7nm high-performance client CPU. We observe an average Time-until-hotspot (TUH) that is 2× shorter than in its 14nm cousin for many SPEC2006 benchmarks, and we observe TUH varies by up to 2 orders of magnitude between different benchmarks. The first hotspot arises after only 0.2 ms. We then use HotGauge to compare hotspot severity across different floorplans, and we observe that floorplanning-based hotspot mitigation techniques like area scaling are inadequate. To enable the broader community to conduct architecture-level hotspot mitigation research, HotGauge, along with all models used in the case study in this work, is publicly available at https://github.com/TuftsCompArchLab/HotGauge and https://doi.org/10.5281/zenodo.5523504.

I. Introduction

Since the end of the Dennard scaling era, power density has increased exponentially with each process node generation. This fact, coupled with the trend of cramming more system functionality into smaller and smaller form factors, has led to the exacerbation of thermal hotspots—or areas of the chip with abnormally high heat flux. These new, "advanced" hotspots (shown in Fig. 1) are becoming an imminent challenge in the way of realizing the benefits of next generation process technology. Advanced hotspots not only have high absolute temperatures, but they are also fast, localized, highly non-uniform, and application dependent. Fig. 2 shows the distribution of the amount by which die temperature changes over 200 µs intervals for 14nm compared to 7nm. The 7nm die is worse in two ways. First, the peak change in temperature is greater, resulting in faster temperature spikes. Second, the variance in temperature deltas is wider, indicating the potential for large temperature deltas like those in Fig. 1. Lastly, all of these changes take place over only 200 µs, indicating that techniques to mitigate hotspots will need to be even more aggressive than they previously were, resulting in the need for increased guard-bands at the cost of dramatically decreased performance.

§Authors contributed equally to this work.

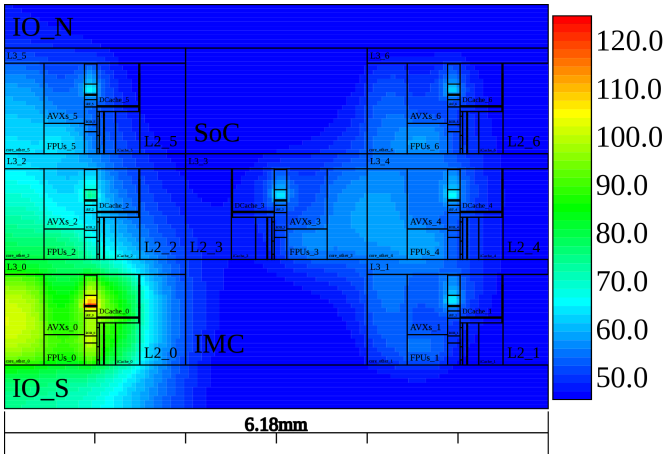
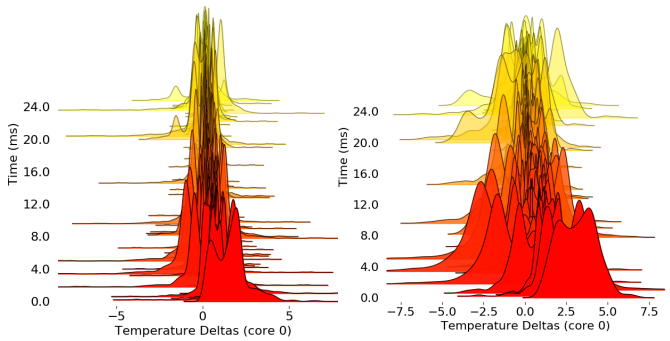


Fig. 1: Advanced hotspot in a 7nm client processor similar to Intel Skylake with untenable temperature gradients. Hot functional units surpass 120°C while nearby units within 200 µm remain 30 degrees cooler.

Thermal regulation is far from a new problem in system design. Researchers across the computing stack have been developing thermal regulation techniques over the last few decades. In the physical cooling domain—heat sinks, thermal interface materials, and liquid cooling, among other techniques, have been proposed to tackle hotspots [9], [16], [22], [23], [25], [45], [46]. These solutions avoid the expensive state-of-the-art chip design process, and they are able to work in concert with solutions at different levels of the computing stack. In the design implementation domain, floorplanning and standard cell placement methods have been proposed to mitigate hotspots [15], [28], [33], [36], [43]. These techniques may have higher design complexity (e.g., re-optimizing EDA tools/flows), but they allow for more fine-grained control which can reduce area costs of the mitigation. Some other mitigation techniques that have been proposed utilize additional OS routines and compiler optimizations [34]. Previously, techniques such as these could be employed in concert to sufficiently quell hotspots. For example, a system could utilize fans, liquid cooling, power-gating, and dynamic voltage and frequency scaling (DVFS) [8], [9], [15], [16], [19], [22], [23], [25], [27], [28], [33], [34], [36], [43], [45]–[47] to keep temperature at a manageable level with minimal impact on performance.

Given the severity of advanced hotspots, these previous techniques no longer suffice. Most of the techniques either do not scale well to ultra-short time constants, or they are not workload-aware or



(a) 14nm (b) 7nm

Fig. 2: Distribution of temperature deltas over 200 µs in the active core while running single-threaded workload for (a) 14nm and (b) 7nm. The thermal grid has a 100 µm resolution.

thermal-aware. As we will show, the frequency of fast, application-dependent hotspots greater than 25°C hotter than neighboring units is rapidly increasing. This threatens future microprocessor performance and technology scaling unless new thermal regulation techniques are developed. Given the nature of advanced hotspots, system designers and architects need to play a more active role in solving this problem by quantifying how a particular system, application or hardware solution impacts the frequency and severity of hotspots. The systems community needs new techniques that adapt to how workloads exercise the processor (or sub-function on-chip) and also the changing, non-uniform thermal state of the die. Fine-grained, architecture-level techniques are needed alongside the traditional techniques to adequately mitigate modern and next generation hotspots. New metrics are needed that formally define hotspots and their severity so that these new architecture techniques can be properly evaluated. Finally, new end-to-end methodologies and tools should be released to energize and empower the research community to tackle hotspots again.

In this work, we present HotGauge: a new methodology for rapid and flexible characterization of the hotspot behavior of modern and next generation processors. HotGauge comes with new metrics to enable meaningful comparison of hotspot severity across different workloads and floorplans. These metrics include: a formal definition of a hotspot, a maximum localized temperature difference (MLTD) metric, a time-until-hotspot (TUH) metric, and an overall hotspot severity (sev) metric. Throughout the rest of the work, we demonstrate the utility of HotGauge with a case study on the hotspot behavior of a modern client CPU and to perform initial architecture-level mitigation studies. HotGauge and all associated models used in this work are publicly available on GitHub and archived on Zeondo.

II. Background and Motivation

HotGauge was developed to respond to an important and pressing problem facing the computing industry: the increasing severity of hotspots. This section describes these hotspots and what the research community needs to address them.

A. Power Density: Localized, Application-Dependant, and Rapidly Increasing

For many decades, Dennard scaling held true [14]. But its loss has resulted in an exponential increase in power density as process nodes scale. We test this phenomenon using two SPEC2006 workloads: bzip2 and gcc. The single-threaded version of each workload was run on a single core of a general desktop processor similar to Intel's Skylake processor and performance and power were measured in the case of different process nodes. We use a V-f point where max power is expected, 1.4 V and 5 GHz, respectively. This is reflective of turbo boost [21], [26], [50].

We observe that total power decreases approximately linearly with each new process generation; however, area reduction is roughly 50% each generation. Unlike during the era of Dennard Scaling—where power density would remain constant, the scaling effects in modern nodes result in increased power density with each process generation. Critically, we observe power density greater than 8 W/mm^2 running bzip2 with 1 thread. This power density is approximately 2× the power density of what could have been expected if Dennard scaling were still in effect. In modern technology nodes—such as 14nm, 10nm, and 7nm— this in turn has greatly increased differences in temperature uniformity on chip, as shown in Figure 1.

These hotspots are local, and appear very quickly as we show in Section IV. Hotspots vary based on how an application uses certain functional units and the current thermal state of the processor. A hotspot is significantly warmer than the silicon area around it, so more coarse-grained techniques that reduce total core power density such as DVFS do not directly address the thermal difference between units and are simply too slow to respond.

B. Lack of a Formal Hotspot Definition

We argue that hotspots should be measured and mitigated in the design process with the same rigor as performance and power/energy. To accomplish this, the community first needs a formal definition of a hotspot, metrics for comparison, and the ability to measure them, to allow for comparison across different architecture proposals.

There are a variety of different hotspot definitions used in the literature [9], [22], [23], [33], [36]. For example, in [9], [22], one of the works uses a hotspot definition of 1 kW/cm^2 while the other uses 680 W/cm^2. Further, these definitions are not tied explicitly to real phenomena, like reliability or timing issues. Furthermore, many works that discuss hotspots do not report any formal definition, such as HotSpot [39], [42] and an updated McPAT [48]. In this work—with the help of our industry collaborators—we develop a rigorous hotspot definition which explicitly addresses the real-world concerns associated with hotspots, i.e., reliability and timing.

Once a hotspot is defined, architects need ways to measure hotspots and the severity between workloads and architectures. There are currently no known methods that summarize hotspot characteristics which are sufficient for modern hotspots, and that allow for easy and quantifiable comparison. We are the first to propose the necessary methods and metrics, and the ones we develop are parameterizable so that they can be adapted to different architectures. Our hotspot definition, metrics, and methods are described in Section III.

C. Limitations of Existing Simulation Methodologies

Simulation in particular plays a key role in architecture-level hotspot research and early-stage design. The entire tool-chain needs to work together; the thermal simulator should interface with the performance and power simulation toolchain. This is because the thermal state of the chip will impact the performance and power of the system, e.g., increased temperature will increase leakage power. Additionally, a hotspot-aware mitigation technique will need to modify microarchitecture state based on the thermal state.

As far as we are aware, there is only one published tool chain which combines performance, power and thermal simulation tools:

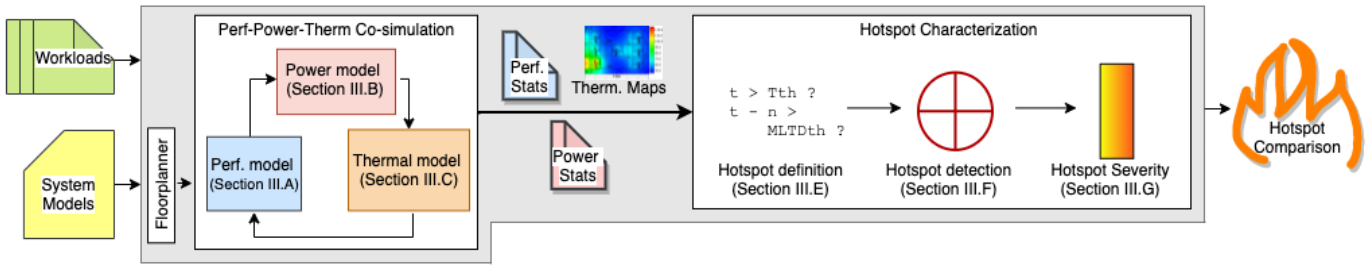


Fig. 3: Overview of HotGauge.

HotSniper [30]. Here we highlight a few of the prohibitive drawbacks of HotSniper for modern hotspot research. First, the newer instruction-window-centric Sniper simulation model (also known as the ROB model) which has been shown to be more accurate for microarchitecture studies [13] is not used, which results in insufficient accuracy to model the fine granularity of modern hotspots. Second, HotSniper models an antiquated 45nm Gainestown processor, and uses Hotspot-6.0 [42], which has limited configurable layers, no socket I/O interface for co-simulation, and is less actively developed than other thermal models, like 3D-ICE [40], [41]. Third, HotSniper models only 6 units per core while modern processors can have upwards of 50 units.

While we plan to release the models used for this CPU case study along with HotGauge, the HotGauge framework is flexible and can be adapted to model other architectures.

III. HotGauge

HotGauge is a comprehensive methodology for architecture level hotspot research; it provides rapid, flexible simulation and characterization of hotspot behavior in modern and next-generation processors. An overview of HotGauge is shown in Figure 3. HotGauge takes, as input, the system models and workloads to be evaluated for hotspots. The first major task is *perf-power-therm co-simulation* which integrates performance, power, and thermal models to perform rapid end-to-end thermal simulation. These models should be highly flexible and extensible to model different architectures, floorplans and system designs. The next major task is *hotspot characterization*. This novel methodology includes a rigorous definition of a hotspot, an automated method for detecting them, and a quantitative way to classify hotspot severity which enables comparisons between floorplans.

In this paper, we demonstrate the capabilities of HotGauge through a case study of a next generation high-performance client CPU similar to an Intel Skylake. However, HotGauge is system-agnostic and is also capable of being used for any processor such as GPU or ML accelerators, like the TPU, if provided with a power and performance model. The exact microarchitecture parameters for the CPU model are listed in Table I. It is an out-of-order (OoO) architecture with 7 cores[1]. Simultaneous multi-threading (SMT) has been modeled with 2 threads/core. Each core includes a 224-entry reorder buffer with 72 and 56 entries in the load and store queue, respectively. The on-chip memory consists of private L1 and L2 caches with 32KiB and 512KiB, respectively. The shared L3 cache has a capacity of 512KiB in a 4-way set-associative configuration with 64B lines. We use approximately a 5mm^2 die where each core has an aspect ratio of 3×2. For frequency and process node, we select 5GHz (representing operating points with frequency-bound serial performance, e.g. turbo boost) and model three different process nodes (14nm, 10nm, 7nm) to analyze the effect of transistor scaling on hotspot severity. SPEC2006 [17] workloads are used in this work.

[1]The use of 7 cores is just used for the prototype we have created for detailed thermal analysis of different core permutations and relative-positions; however, any floorplan is possible.

A. Performance Model

The base performance model that we use in this case study is the Pin-based, x86 multi-core simulator, Sniper [12], which is used extensively both in the research community and in industry. Sniper is configured to use the instruction-window-centric model [13] as it has been demonstrated to be the most accurate. When choosing the performance model for the toolchain, it is important to select one which has an interface for co-simulation given that thermal events will have a dynamic effect on performance. Sniper meets this requirement. Each workload is run under the performance simulator for a fixed number of instructions after an appropriate cache warm-up. We simulate each workload's region of interest(*roi*) for 200M instructions, using a time step of 1M cycles. This is sufficient for transient thermal simulations since we are interested in hotspot behavior which has very short time scales (unlike thermal steady state). Exact simulation duration depends on the total length of a workload as well as on the lengths of the regions of interest; however, cache warm-up is always performed. Performance stats are then post-processed to meet the input specifications of the power model and fed into the power simulator.

TABLE I: The client CPU microarchitecture model used in this work

CPU Microarchitecture Parameters	
Process node [nm]	14, 10, 7
Cores	7
Core area [mm^2]	5, 2.5, 1.25
Frequency	5 GHz
SMT	2
ROB entries	224
LQ entries	72
SQ entries	56
Scheduler entries	97
L1I $	Private, 32 KiB
L1D $	Private, 32 KiB
L2 $	Private, 512 KiB
L3 $	Shared ring, 16 MiB

B. Power Model

The base power modeling tool we use for this study is McPAT [24] v1.2. We run it in the highest granularity setting at each time step to generate power statistics. We use a frequency of 5 GHz and a voltage of 1.4 V to represent turbo boost. Previously, the lowest supported technology node in the most recent McPAT was 22nm. We add novel

modifications to McPAT to model advanced technology nodes. Using standard transistor scaling trends, we tune leakage, dynamic power, and area to model 14nm, 10nm, and 7nm nodes. We use 50% area scaling node to node, and a 20% decrease in C_{dyn} [7], [37], [49]. It is even possible to scale beyond 7nm if desired.

C. Thermal Model

The base thermal model we use in this study is an open source tool called 3D-ICE, version 3.0.0 [40], [41]. 3D-ICE is a compact transient thermal model (CTTM) made specifically to model 3D ICs. Previous iterations of the perf-power-therm simulation toolflow used for this case study featured Hotspot as the thermal model; however, it was replaced with 3D-ICE due to the limitations outlined in Section II. 3D-ICE allows for both steady state and transient thermal simulation, and it has been validated with real prototype experiments.

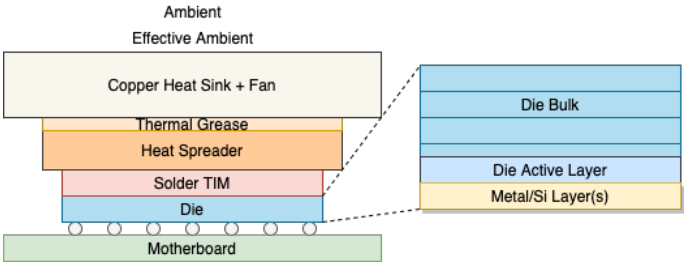


Fig. 4: Layers in the thermal model used in this case study. Note: solder balls and motherboard are not modeled and are for reference only.

TABLE II: Thermal configuration parameters for the thermal model layers used in this case study. The heatsink model used was included with 3D-ICE and models a HS483-ND heatsink with a P14752-ND fan spinning at 6000 rpm.

Layer	**Thermal Conductivity** [W/μm K]	**Volumetric Heat Capacity** [J/μm^3 K]	**height** [μm]
Thermal grease	0.04e-4	3.376e-12	30
Copper (heat spreader)	3.9e-4	3.376e-12	3e3
Solder TIM	0.25e-4	1.628e-12	200
Silicon (IC wafer)	1.20e-4	1.651e-12	380

The thermal stack used in this work includes the Silicon IC, a solder Thermal Interface Material (TIM), a heat-spreader, thermal grease, and a heatsink as shown in Figure 4. The IC is then further divided in the vertical direction between the active layer and the bulk of the die. This was done to increase the resolution and accuracy of the thermal model within the die, which is essential for realistically modeling hotspots. Thermal simulations without these added layers resulted in scenarios where heat spread more horizontally and less vertically than expected. This was due to the discrepancy between the node distance in the intra-die plane (i.e. the resolution of the thermal images shown) and the vertical node distance (between layers). The configuration parameters for the thermal stack are displayed in Table II.

To enhance the realism of these models and tools, we make our own additions: thermal and floorplan models for AVX512, System Agent/SoC, Memory Controller (IMC) and I/O. We also implement non-uniform temperature initialization of the thermal stack in 3D-ICE to model the fact that CPUs have other workloads running on the system (e.g., background tasks, OS tasks, and recently context switched workloads). The inputs to the thermal model include the floorplan, a power trace for each flooorplan element, and a stack description file. The floorplan contains all of the functional units in the McPAT output (run in it's most granular setting) as well as the additional units added in our model as shown in Figure 5.

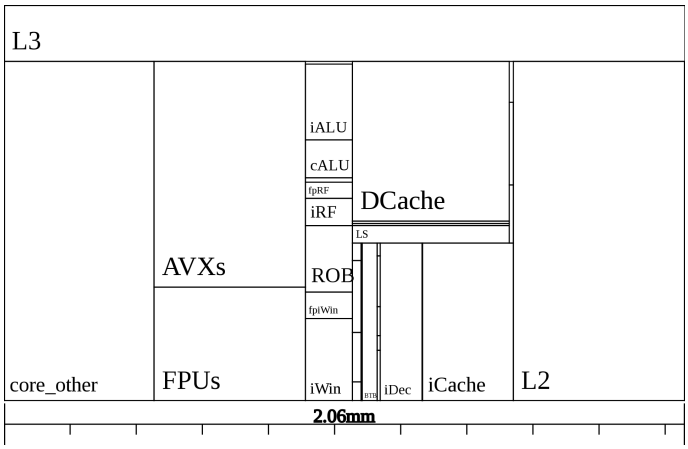


Fig. 5: The floorplan of each core used in this case study. The floorplan is inspired by the Intel Skylake processor.

D. Validation

HotGauge is flexible and capable of modeling different microprocessor and thermal configurations. We independently validate both our power model and thermal stack. For the study presented in this work, we calibrated the Skylake-proxy processor model with support from industry collaborators. This includes calibrating both C_{dyn} (effective dynamic switching capacitance), the power reported by McPAT for the SPEC2006 benchmarks, and the thermal stack. This is to ensure that the shape of the hotspot problem is being modeled accurately and matches in-house models for 14nm, 10nm and 7nm.

Power Model Validation. We performed additional validation which can be publicly shared that demonstrates the accuracy of the 14nm and 10nm models compared to commercially available processors. We used the Intel Thermal Analysis Tool (TAT) to read Intel's IA core power plane register which provides a direct measurement of power consumption on a real Intel processor. We used an Intel Core i5 10310U processor which is a 14nm mobile part and a 11th gen Intel Core i7-1165G7 which is a 10nm part with SuperFin transistors. C_{dyn} is our validation parameter since it is invariant to voltage and frequency. Single core C_{dyn} is also consistent across all form factors (mobile, server, desktop) with the same microarchitecture generation (e.g. Skylake). It is computed from measured power, voltage, and frequency after leakage power was isolated. A summary of the results for the non-fortran[2] SPEC2006 workloads is shown in Table III. We generally observe reasonable error for the validation set. Average error for the 14nm model and 10nm model is 11% and 20%, respectively. The increased error for 10nm is expected as it is a different microarchitecture with increased resources. Prior work, with privileged access

[2]The non-fortran workloads were used simply to avoid backporting old fortran libraries on new Ubuntu systems.

to Power7 models, was able to come within 10-20% error [48]. It is not possible to capture all circuit design characteristics in a high-level power model such as McPAT and more importantly it is the general scaling trends that we are trying to capture and not the specifics of a particular Intel core.

TABLE III: C_{dyn} percent error of SPEC validation set for this case study. The real silicon includes an Intel Core i5 10310U processor, which is a 14nm mobile part and a 11th gen Intel Core i7-1165G7, which is a 10nm SuperFin part.

	C_{dyn} [nF]					
	14nm Si	model	error	10nm Si	model	error
bzip2	1.33	1.36	+2%	1.32	1.17	-11%
gcc	1.51	1.30	-14%	1.80	1.13	-37%
omnetpp	1.16	1.33	+15%	0.99	1.16	+17%
povray	1.87	1.62	-13%	1.87	1.36	-27%
hmmer	1.52	1.65	+9%	1.49	1.38	-7%
abs. avg.	-	-	**11%**	-	-	**20%**

Thermal Stack Validation. We also evaluate our thermal stack design by computing the overall thermal resistance from the IC junction layer(j) to ambient(a), $\Psi_{j,a}$, which is measured in units of $^{\circ}C$/W. Intuitively, this metric describes the amount of temperature increase in the IC active layer—relative to ambient—per Watt of power consumed. Additionally, we estimate the Thermal Design Power (TDP) of each of the ICs used in this work when cooled by our thermal stack. For this calculation, we assume a local ambient temperature of $40^{\circ}C$ and maximum operating temperature of $100^{\circ}C$, resulting in a thermal budget of $60^{\circ}C$. To compute the TDP for each part, we simply divide this thermal budget by Psi. The values of $\Psi_{j,a}$ and TDP for each technology node are shown in Table IV. The values of TDP range from 43-63W, which is consistent with a high-performance/gaming laptop or a small form-factor desktop that may include a processor like the Intel H-series, which has a TDP that is between 45 and 65W [3], [4]. It should also be noted that the value for Ψ increases with advanced technology nodes because, although the heatsink is the same, the IC is smaller[3], resulting in a higher thermal resistance between the IC and the heatsink.

TABLE IV: Ψ and TDP for thermal model used in this case study. TDP calculation assumes a thermal budget of $60^{\circ}C$.

	Process Node [nm]		
	14	10	7
Ψ [$^{\circ}C$/W]	0.96	1.13	1.40
TDP [W]	63	53	43

E. Hotspot Definition

To study the behavior of hotspots, it is first necessary to define what exactly constitutes a hotspot. The goal is to capture all phenomena which will cause either performance loss or reliability problems [5], [11], [20], [29], [31], [32], [35], [38]. Using insights from industry, we therefore classify a temperature event as a *hotspot* using two main heuristics: absolute temperature and maximum localized temperature differential (MLTD). Absolute temperature negatively affects reliability and, in fact, affects transistors and interconnect differently. Large MLTD, on the other hand, affects performance by impacting the safe clock timing margin. Localized temperature differentials arises due to (1) the different ways in which temperature affects transistors vs interconnect and (2) the difference in activity between neighboring transistors, sub-units, or functional units (depending on the granularity of interest). When an unchecked large MLTD occurs within the same functional unit or sub-unit, it is possible for it to cause timing violations. If the temperature of a point on the die exceeds a specified threshold and the maximum temperature differential between that point and neighboring points—within some specified radius—exceeds a specified threshold, then that point is defined as a hotspot. This is concretely defined in Definition 1 below.

Definition 1 (Thermal Hotspot): Let T denote the set of all temperature values on the die; let N denote the set of all temperature values within radius r of a temperature t in T. Then for some absolute temperature threshold, T_{th}, and MLTD threshold $MLTD_{th}$, t is defined to be a hotspot *iff* $t > T_{th}$ and $t - n > MLTD_{th}$ for any n in N.

The selection of values for temperature threshold, radius, and MLTD depend on the granularity of interest as well as the particular study. Modern hotspots can occur anywhere on the scale of transistors up to the scale of functional-units; so, radius should be chosen accordingly. In this work, we focus on hotspots between sub-units, so we choose a radius of 1mm, which is roughly the maximum distance that can be covered in one clock cycle in a modern high performance processor.

The reason we keep radius fixed across technology nodes is to capture the fact that global wires do not scale very well across technology nodes and die-size has historically remained constant as more logic is placed on core [6], [10], [18], [44]. In light of these trends, the length of the critical timing paths will be approximately the same across nodes. In the case of absolute temperature threshold and MLTD threshold, we select values of 80°C and 25°C, respectively. This is to represent conditions which have been observed to cause problems for reliability and timing within functional units. Maximum junction temperature of a typical desktop processor is around 105°C [1], [2]; therefore, these thresholds correspond to an occurrence of reaching the maximum temperature of the chip, resulting in throttling.

In general, the suitable values of these parameters are determined by a variety of characteristics of a given manufacturing process and specific IC. The length of the critical timing path will determine the length over which MLTD should be computed. The threshold values for temperature should be chosen based on the maximum acceptable temperature plus any additional guard-bands to account for thermal-sensor characteristics, including distance from the heat source and sensing/response times. For example, the threshold can be based on T-Junction (minus a guardband)—as in this work The MLTD threshold should be chosen based on a combination of the effect of changing temperature on timing and the timing slack, which, together, determine the acceptable MLTD; i.e. when the difference in temperature creates a larger difference in timing than the timing slack can accommodate.

HotGauge is a tunable framework so this definition can be tuned by the designer to match their system conditions. For example, if the timing paths were much shorter, a user would reduce the 1mm radius; if the system needed more time to respond to fast transients the temperature gradient would be reduced from 25C; and if a system had logic that was more sensitive to temperature in the cooling path

[3]In this work, we study the effect of advancing technology node rather than micro-architectural changes, so we keep the floorplan layout and processor composition consistent across nodes

e.g. stacked DRAM, then the temperature would decrease from 80C to 70C depending on the DRAM tech.

F. Hotspot Detection

Using the rigorous definition from the previous subsection, we automate hotspot identification and integrate it into HotGauge. The naive version of a hotspot detection algorithm would be to iterate over every thermal pixel, compute the MLTD within some radius, and then check thresholds. While this implementation is robust, it is also very inefficient and computationally expensive. If a temperature, $t_{n,m}$, on the IC is greater than T_{th} and $MLTD_{n,m}$ is above $MLTD_{th}$ (see Definition 1), it is very likely for that condition to also hold for pixels adjacent to, and even nearby, $t_{n,m}$. To mitigate this inefficiency, we developed a multi-step approach that first identifies candidate locations and then computes MLTD at those locations only. In our algorithm (shown in Figure 6), a hotspot candidate is defined as a local maxima in temperature in both the x and y dimensions. This drastically reduces the computational load of our algorithm while ensuring that the worst possible hotspots are still considered. While nearby locations may still be classified as hotspots, the local maxima can be considered as the true location of the hotspot. Given this list of hotspot candidates, hotspots are declared to be present where both t and MLTD are above their respective thresholds.

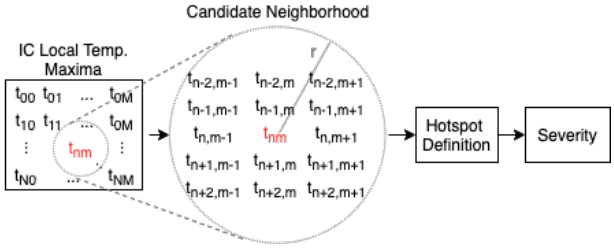


Fig. 6: Automated hotspot detection algorithm.

G. Hotspot Severity

While it is important to be able to identify individual hotspots, designers also need methods and metrics to compare hotspot behavior across the entire chip or functional unit. Such a metric is essential to evaluating and comparing the efficacy of hotspot mitigation strategies. We develop such a metric and demonstrate its use in evaluating mitigation techniques in Section IV. We define the range of the metric to be between 0 and 1, where a value of 0 indicates conditions where there are no hotspot related issues and where a value of 1 indicates conditions where the chip will experience immediate errors, crashes, or permanent damage. We further define that the value should be 0.5 under conditions where immediate mitigation steps are required in order to avoid hotspots. This provides a critical point of reference for those developing and using the metric, as well as balances the range of values associated with 1) highly concerning hotspots (>0.5) and 2) regions that are not yet of concern, while still providing a quantitative value that can be used to quantify increasing or decreasing potential for concern over time. In the regions between these defined values, the severity metric should take on a continuous range of values.

In order to get this desired behaviour, we first define a parameterized sigmoid function with the form shown below in Equation 1.

$$\sigma(x_o, y_o, s, a) = \frac{a}{1 + e^{-s(x - x_o)}} + y_o \tag{1}$$

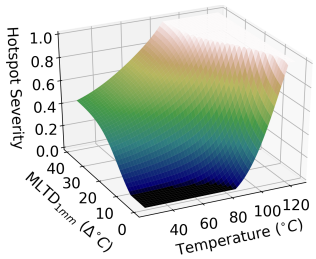


Fig. 7: Hotspot severity metric from Equation 2. This metric is tuned for high-speed CPU-like circuits without DRAM in the thermal stack. Values at 1 indicate that an error or permanent device damage is imminent; 0.5 or above indicates mitigation is necessary.

This parameterized sigmoid function was chosen because it enables smooth interpolation between low and high values, and has other parameters that can be adapted as needed in order to match the desired constraints. In the parameterized version of the equation, x_o is the x offset, y_o is the y offset, s is related to the slope of the sigmoid, and a is the overall amplitude of the sigmoid. We then define the hotspot severity metric using three parameterized sigmoid functions as defined below in Equation 2 and plotted in Figure 7.

$$sev(x,y) \;=\; \sigma_{df}(T_{x,y}) + \sigma_M(MLTD_{x,y}) * \sigma_T(T_{x,y}) \tag{2}$$

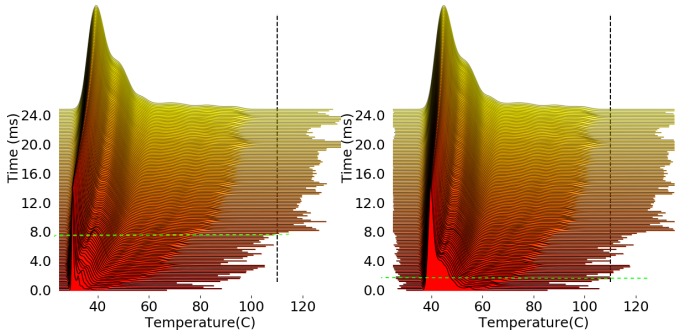
where:
$\sigma_{df} = \sigma(115, 0, 0.2, 2)$
$\sigma_M \;= \sigma(15, -0.25, 0.2, 1.25)$
$\sigma_T \;= \sigma(60, 0.35, 0.05, 0.65)$

In addition to meeting the above specifications—including being clipped to be in the range of 0 to 1—this metric is broken down into different components that can be tailored to new process technologies and thermal environments, where the desired operating conditions may result in different considerations. The first term (σ_{df}) defines where *device-failure* is imminent based solely on the temperature of the IC. For this work, we use a sigmoid centered at $115°C$ with an amplitude of 2 in order to saturate to 1 at $115°C$. This value reflects the junction temperature of modern processors without a guardband [1], [2].

The second term in our metric is the product of σ_M and σ_T, which are the marginal contributions of MLTD and T for timing issues. It is necessary to consider both MLTD and T simultaneously because of the non-linear relationship between timing and temperature, which happens due to the opposite effects on timing that temperature has on logical circuits relative to global interconnects. In general, these values need to be set such that the value of the severity metric is related to the probability that a timing related failure will occur, i.e. how close the differences in timing come to surpassing the available timing slack in a path. While each of these factors is heavily dependant on manufacturing process characteristics and IC implementation details, the metric is flexible such that its parameters can be tuned to fit a given use case. The parameters here, a contribution for the case study, are fitted from industry data and insight.

IV. Results and Analysis

In the subsequent analysis, we unveil the dangerous state of the hotspot problem for modern and next-generation client CPUs. To preview, absolute temperature is high enough that existing mitigation techniques will need to become so aggressive to the point where they will result in dramatic performance loss. Furthermore, hotspots arise 2× faster in 7nm than in 14nm for a variety of SPEC2006 benchmarks, indicating that current and future hotspot control and mitigation techniques will need to be sensitive to even finer time constants. We discover that MLTD measured at a 1mm distance is worse for more advanced nodes, which forecasts challenges for global wires and timing paths. When we look at hotspots across benchmarks, we see that TUH variation is greater than 2 orders of magnitude between different SPEC2006 benchmarks in 7nm (anywhere from 0.2 ms to 150 ms). We study the effect on TUH that core-placement has on single core workloads and observe that TUH can vary by over an order of magnitude from core to core for some workloads in 7nm. Furthermore, when considering the current thermal state of the die, we notice TUH of the 7nm processor varies by greater than an order of magnitude based on which core the single-core workload is running on for various workloads. We can, unfortunately, conclude that TUH in 7nm is so low that more aggressive throttling will be required which will have a certain impact on performance in 7nm and beyond.



(a) From Ambient (b) After Idle Warmup

Fig. 8: Temperature distribution (x-axis) over time (y-axis) for gcc in 7nm after (a) no warmup (from ambient) and (b) an idle warmup. Note: the height of each distribution is normalized and color is added in order to aide visualization. Dotted green lines show the time when peak temperature surpasses 110°C.

A. Absolute Temperature

Technology Scaling. In order to characterize how the behaviour of processors on newer technology nodes compare with older ones, we first consider high level thermal behavior, namely absolute temperature, of the chip. In order to isolate the effect of technology scaling, the same floorplan layout is used across technology nodes, and the area is scaled to match the expected area of the processor in that technology node. First, we summarize general trends we observed (raw data could not be included due to space limitations): for example, for the workload gcc (from ambient), the 7nm node mean temperature increases faster—*7nm reaches 35°C roughly 5× faster than 14nm*—indicating that hotspots could potentially develop in shorter time periods on newer nodes (we observe similar behavior for other benchmarks). Local heating caused by hotspots is a concern when we see fast heating because the heat spreader and heat sink, which are designed to disperse thermal energy, have longer thermal time constants. Furthermore, *the maximum temperature of the 7nm node surpasses 90°C roughly 3× faster than the 14nm node*—which suggests that the hotspots on the 7nm node have the potential to surpass even higher thermal gradient thresholds.

Warm Up. To get a more complete picture of the thermal profile of the die, we look at all temperatures across the die as a function of initial thermal conditions for the same example benchmark, gcc, and the most aggressive technology node, 7nm (again, we observe similar behavior for other benchmarks). Figure 8 shows these visualizations. Each horizontal curve represents a histogram of temperature for all nodes on the die for a particular time step during the simulation. The continuous peak in the z dimension across time (which looks like a tall wave) indicates a larger proportion of nodes in the die corresponding to that temperature.

After an idle warmup, not only is there increased variation in temperature across the IC relative to the simulation from cold, a threshold of 110°C is surpassed more than 4× faster. Temperatures in this range could result in circuit failure, and are thus avoided by techniques such as throttling the CPU, which is costly in terms of performance. These findings indicate that initial thermal state will have significant impact on thermal behaviour, and must be accurately modeled in order to develop mitigation techniques.

Additionally, these trends have two major implications for on die temperature sensors. First, in order to respond to the fast temperature changes that are observed, thermal sensors will have to have correspondingly fast response times. Second, they must also be placed in regions of the die which are more likely to experience extreme temperatures, as even when some of the die surpasses 120°C, some of the die may still be close to ambient temperature, even if the processor has been on prior to running the workload.

B. Maximum Localized Temperature Difference

While an increase in absolute temperate demonstrates a high level risk for hotspots, it does not guarantee the presence of hotspots (for example, the die could heat up quickly but in a uniform manner). Now we look at the Maximum Localized Temperature Difference (MLTD) over time in order to understand the difference in hotspot magnitude across process node generations. Figure 9 shows this analysis for a different example workload, gobmk, after initializing the thermal stack with an idle background task thermal warm-up trace (we see similar trends for other workloads). MLTD is computed within a 1mm radius. For the first 20ms of the workload, the MLTD of the 7nm processor is approximately 2× that of the 14nm part, and peaks at almost 70°C whereas the 14nm part peaks at under 60°C.

Core to Core Variation. Figure 9 also shows how the location of the core running the workload affects the MLTD of the IC. Intuitively, the activity of neighboring floorplan elements may have an effect on the thermal characteristics of the core under test. Notably, this effect does not manifest itself as much in the case of 14nm; however, *in the case of 7nm, we observe a significant effect based on orientation of the core running the workload.* The MLTD in the 7nm processor is the highest for cores 0, 2, and 5—which all lie on the left side of the die— lowest for cores 1, 4, and 6—which all lie on the right side of the die, and falls in the middle for core number 3 which is in the middle of the die. This is likely due to a combination of thermal edge effects combined with the relative power densities of the units in the processor on each side of the core.

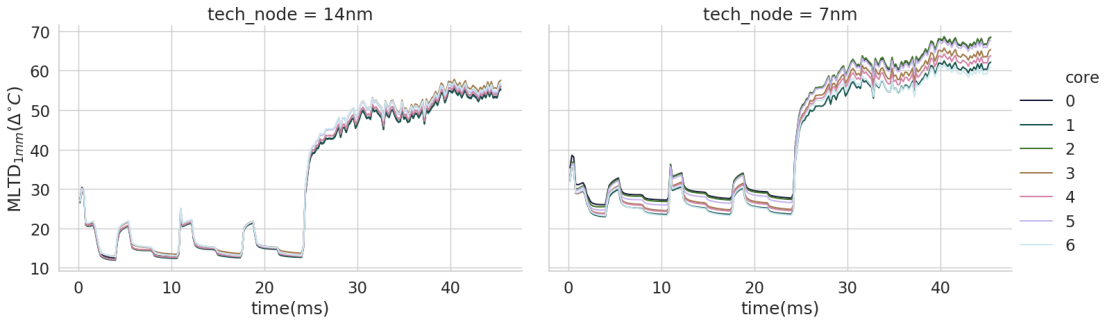


Fig. 9: Maximum localized temperature difference (MLTD) within 1mm radius for single-threaded gobmk after idle warmup.

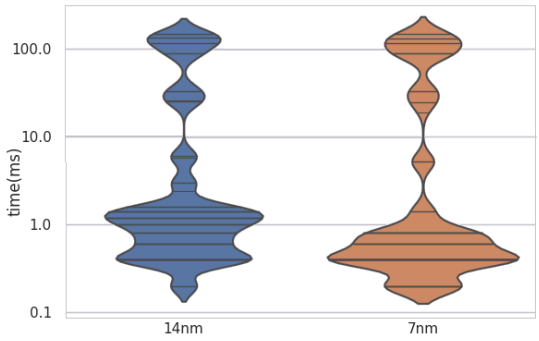


Fig. 10: Time until Hotspot (TUH) (T_{th}=80°C, $MLTD_{th}$=25°C) for SPEC benchmarks after idle warmup. The 5th, 25th and 50th percentiles for 14nm are 0.4ms, 0.6ms, and 1.2ms respectively, and roughly half that for 7nm: 0.2ms, 0.4ms, and 0.6ms.

C. Time Until Hotspot

At this point, we have demonstrated that the increase in absolute temperature and MLTD from one process node to the next is unsustainably high. Another important consideration is the Time Until Hotspot (TUH) which we measure as the time from the start of the application until the first formally defined hotspot appears. This has obvious implications for the time resolution required of on-chip temperature sensors, the speed of heat spreading and cooling, or other mitigation solutions. In this section, we focus on TUH as a function of technology scaling and workload.

Technology Scaling. We first focus on the affect of technology scaling on TUH. Figure 10 shows TUH on a log-scale as a function of technology scaling for the SPEC workloads after an idle warmup. We use our hotspot definition from Definition 1 in Section III-E with T_{th}=80°C and $MLTD_{th}$= 25°C. For the hotspots that occur later in the workloads, e.g. after 5ms have passed, there is little difference in the distribution of TUH across 7nm and 14nm, meaning that TUH is relatively unchanged. Therefore, it is likely that these values of TUH are due more to the overall behaviour of a given workload, namely that it has a sudden and dramatic spike in computational intensity at a certain phase in its execution.

In the region where TUH is under 1ms, however, there are notable differences between 7nm and 14nm. In this region, the 7nm node contains considerably more examples with lower TUH as demonstrated by the widening of the distribution at its lower bound. Consequently, the 5th, 25th, and 50th percentiles for 14nm are all higher than those of (7nm); namely they are are 0.4ms (0.2ms), 0.6ms (0.4ms), and 1.2ms (0.6ms) respectively. Thus, we conclude that for the SPEC workloads, hotspots occur in roughly half the time. This suggests that *more aggressive throttling techniques* will be required, which will lead to performance loss.

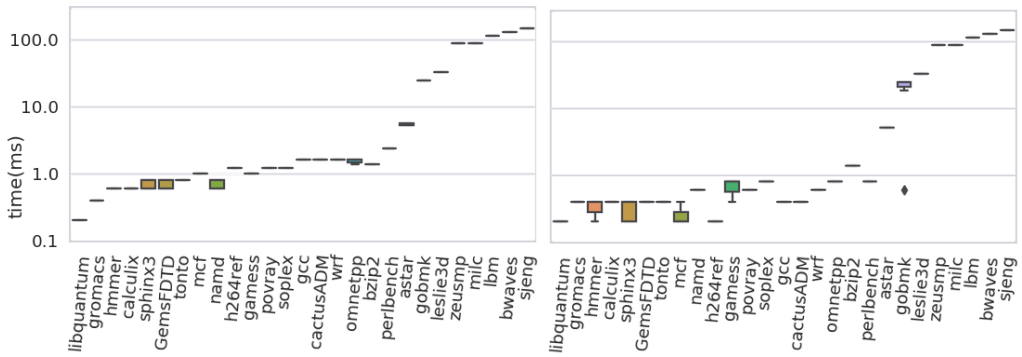
Workload and Warm Up. In addition to variation in TUH across technology node (shorter TUH for more advanced nodes), we observe TUH variation across workloads and initial thermal states. Figure 11 demonstrates this point by showing a box and whisker plot[4] of the TUH for each of the SPEC2006 benchmarks when run on each core (individually) in the 7nm technology node. We observe *greater than 2 orders of magnitude difference in TUH across SPEC2006 benchmarks*—anywhere from 0.2 ms to 150 ms. When considering whether the processor starts from cold or from an idle warmup, TUH is not impacted significantly for most benchmarks. However, for some workloads, such as gobmk and namd, TUH does in fact vary across warmup scenario. In the case of gobmk, TUH is *significantly* lower after an idle warmup, but only for a few cores, while the other TUH values remain relatively unchanged. On the other hand, namd actually develops a hotspot *more quickly* when it executes from cold, which highlights the reality that lower and more uniform initial temperatures can actually increase MLTD, and thereby decrease TUH.

Core to Core Variation. As previously mentioned, the core that is assigned the single threaded workload is swept for each benchmark. In some cases, the location of the core has little effect TUH, e.g., libquantum, gromacs, calculix, GemsFDTD, etc.; however, in roughly 20% of the benchmarks we observe wider distributions of TUH, indicating core-dependant values of TUH. In the case of gobmk after an idle warmup, we observe an *order of magnitude difference in lower and upper quartile of TUH*. In light of this, dynamic mitigation techniques will need to be aware of the placement of a thread on the die in order to be effective.

D. Hotspot Locations

Here, we discuss *where* hotspots occur in the processor. Figure 12 shows all the locations where hotspots occur in a core when running SPEC workloads. The majority of hotspots are located in the Complex ALU (cALU), the Floating Point Instruction Window (fpIWin), the Register Access Tables (RATs), the Register Files (RFs), miscelaneous core logic (core_other), and the Reorder Buffer (ROB). Interestingly, we noticed that seemingly innocuous characteristics such as core orientation affect where hotspots occur (not shown). For example, the "core_other" block (which contains miscellaneous logic) only contains

[4]A "box" shows the spread of TUHs between the first and third quartile, and the "whiskers" show the minimum and maximum TUH.



(a) Cold Warmup (b) Idle Warmup

Fig. 11: Time until Hotspot (TUH) (T_{th}=80°C, $MLTD_{th}$=25°C) in 7nm for SPEC after (a) no warmup (from ambient) and (b) idle warmup. Data for each benchmark is aggregated across simulations where the single-threaded workload binary is run on each core individually.

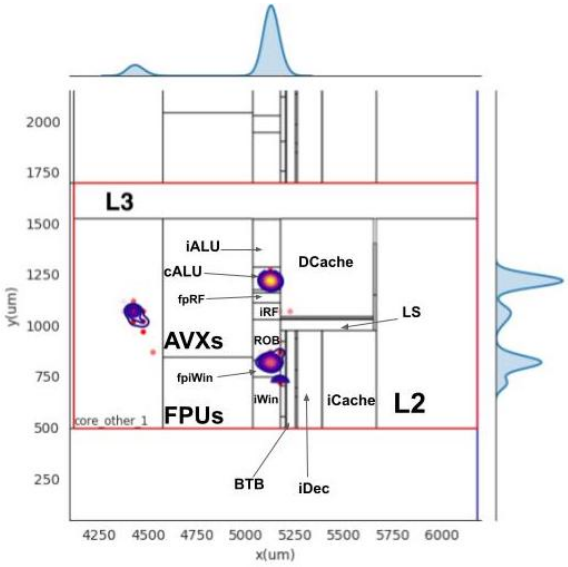


Fig. 12: Locations of hotspots in 7nm aggregated over all single-threaded SPEC benchmarks.

hotspots when it is placed next to another active circuit. It contains no hotspots when on the edge of the die where it could more readily transfer heat to an otherwise unused portion of the heat spreader.

While the primary locations of hotspots for this particular workload were the Complex ALU (cALU), the Floating Point Instruction Window (fpIWin), the Register Access Tables (RATs), the Register Files (RFs), miscelaneous core logic (core_other), and the Reorder Buffer (ROB), other units also manifest hotspots. For example, if AVX-intensive benchmarks were selected, we would see a high volume of hotspots in the AVX unit. This means that mitigation techniques will need to be present *all over the die* for a general purpose system. Understanding the behavior of hotspots—particularly related to specialized units as well as accelerators—is part of our future work plan.

V. Mitigation Case Study

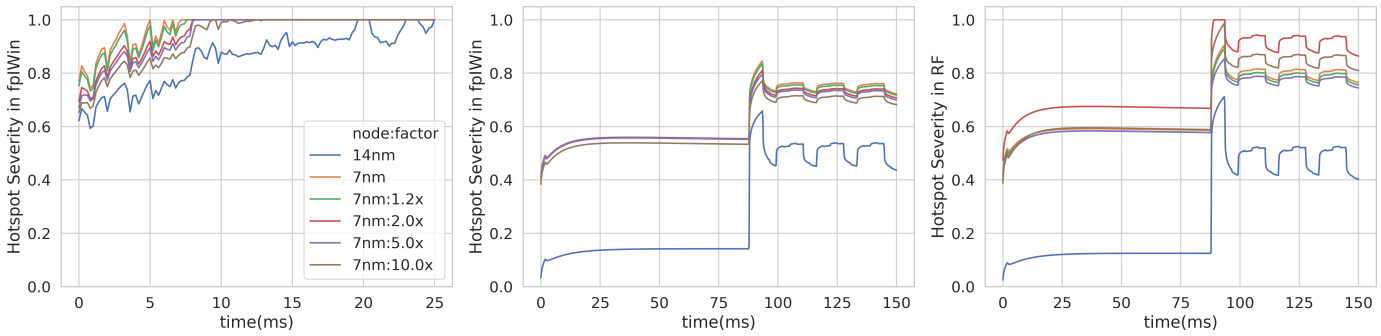
Using HotGauge, it is possible for architects to begin to evaluate architecture-level mitigation techniques by comparing the hotspot severity across multiple designs. While we have ongoing work focused on mitigation, here we highlight a couple of straightforward methods that architects use to mitigate hotspots.

A. Problematic Unit Scaling

As an illustrative case study, we evaluate the effect of scaling the area of a given unit on the hotspot severity *in that unit*. This scenario is similar to the case where a designer desires to reduce the peak hotspot severity within a given unit to be under some critical threshold. We use area scaling as a generic proxy for either 1) reducing unit power, 2) adding idle "white-space" around/within the unit, or 3) a combination of (1) and (2). Each of these has the similar end result of reducing the power density (W/mm^2) of the target unit. For this study, we consider the case where the hotspot severity of the 14nm part is the desired threshold, i.e. the designer knows that thermal problems were manageable in the 14nm part. If the 7nm part can achieve the same levels of hotspot severity as the 14nm baseline, the designer's goal has been achieved.

In order to guide our selection of which units to target, we leverage the data collected in Section IV-D, where units with a larger number of hotspots were identified. For each experiment, we create many new floorplans with scaled versions of the unit under study and then evaluate the hotspot severity for each benchmark in our testing suite. Figure 13 shows the hotspot severity of hot units as a function of time for the 7nm, scaled 7nm, and 14nm floorplans. As Figure 13a shows, the hotspot severity in the Floating Point Instruction Window (fpIWin) while running gcc can be drastically reduced by scaling the size of the fpIWin. However, even when the fpIWin is scaled by a factor of 10×, the hotspot severity in the fpIWin unit of the 7nm processor is still higher than in the 14nm part. Additionally, scaling a single unit like the fpIWin does not always produce the same level of improvement; Figure 13b shows that while running, for example, milc, not only is the hotspot severity in the fpIWin not as high as in gcc, but scaling the size of the fpIWin is far less effective in reducing the hotspot severity level to that of the 14nm processor. In the case of milc, it is actually more effective to scale another unit such as the Register Files (RFs), as shown in Figure 13c. This demonstrates that no single-unit mitigation strategy, even if the scaling is drastic, will effectively reduce hotspot severity across all workloads.

Now we focus our attention to the results across all benchmarks for each unit. We conduct a study using the Register Access Tables



(a) fpIWin Scaling (gcc) (b) fpIWin Scaling (milc) (c) RF Scaling (milc)

Fig. 13: Hotspot severity over time after scaling Floating Point Instruction Window (fpIWin) or Register Files (RFs) for single-threaded gcc and single-threaded milc.

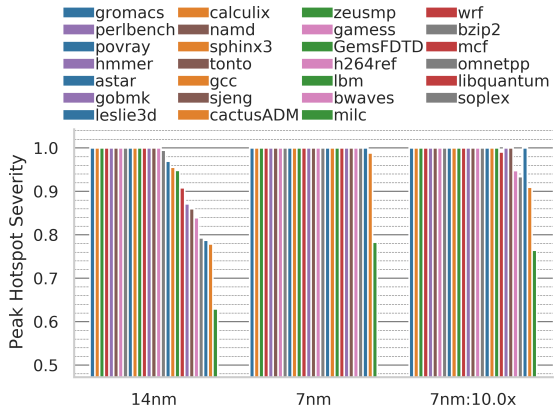


Fig. 14: Max. hotspot severity for SPEC after scaling the Register Access Tables (RATs).

(RATs). Figure 14 shows the max hotspot severity of each benchmark when scaling the Regiser Access Tables (RATs). Again, 14nm severity results are the target for the study and the 7nm results are the initial severities that are to be reduced. The final comparison point is the 7nm floorplan with RATs 10× their normal size. We observe that even after scaling the RATs by 10×, the peak hotspot severity is still higher than the 14nm target, and many workloads still reach a hotspot severity of 1 during the 200M instruction simulation window. This indicates that not only do 7nm processors reach a hotspot severity of 1, the peak severity is still higher than 14nm even after scaling the RATs by 10× which demonstrates that the need for mitigation techniques that can affect multiple units in the chip simultaneously.

B. IC Scaling

As a limit test, we also evaluate the efficacy of reducing the power-density of the entire IC uniformly by evaluating the effect of adding empty area uniformly across the IC, thereby reducing power-density. The limits of this mitigation are evaluated by increasing the area of the 7nm floorplans until they exhibited similar hotspot behaviour to the 14nm tech node. In order to compare hotspot behavior across nodes, we first define $sev(t)$ as the peak hotspot severity on the IC as a function of time. We then compute the Root Mean Square (RMS) of $sev(t)$ for each benchmark in order to summarize the hotspot severity of the IC while the benchmark is running. By using RMS to summarize the $sev(t)$ signal, we not only consider both hotspot duration and magnitude, but we also more heavily weight higher hotspot severity values relative to lower ones, meaning that spending 1ms at severity X is worse than spending 2ms at severity $X/2$. Using this method, we find that in order to reduce the RMS of $sev(t)$ for the 7nm processor to match that of the 14nm processor, the area of the 7nm IC would need to be increased by between 75% and 150%, depending on the benchmark. This demonstrates that static techniques, even if they significantly reduce the power-consumption, have such a large hurdle to overcome, and that, therefore, more targeted and dynamic approaches are necessary.

VI. Conclusion

In this work we introduce a holistic methodology for characterizing hotspots in modern and next generation processors which refer to as HotGauge. HotGauge details new methods and metrics for characterizing and comparing hotspot severity across any next generation processors. This will allow the architecture community to develop architecture level mitigations to work alongside traditional thermal regulation techniques to solve the advanced thermal hotspots which are occuring in modern and next generation processors. To demonstrate the functionality of HotGauge, we use it to perform a case study with a modern high-performance client CPU based on an Intel Skylake. We show that time-until-hotspot (TUH) is 2× faster in 7nm than in 14nm for many workloads, and TUH varies by up to 2 orders of magnitude between different SPEC2006 benchmarks in 7nm, with initial hotspots arising after only 0.2 ms. This suggests that the industry needs more adaptive, architecture-level mitigation techniques to work in concert with conventional techniques. Upon publication of this work, we publicly release HotGauge as well as all related models discussed in this work.

VII. Acknowledgment

This work is supported in part by Google and Intel. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the supporting organizations.

References

[1] "Intel® core™ i7-3615qm processor," https://ark.intel.com/content/www/us/en/ark/products/64900/intel-core-i7-3615qm-processor-6m-cache-up-to-3-30-ghz.html, accessed: 2020-11-20.

[2] "Intel® core™ i7-3770 processor (8m cache, up to 3.90 ghz) product specifications." [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/65719/intel-core-i7-3770-processor-8m-cache-up-to-3-90-ghz.html

[3] "10th gen intel core h-series introduces the world's fastest mobile processor at 5.3 ghz (2020)," https://newsroom.intel.com/news/10th-gen-intel-core-h-series-introduces-worlds-fastest-mobile-processor-5-3-ghz/, Apr 2020.

[4] "Intel® core™ i9-10980hk processor," https://ark.intel.com/content/www/us/en/ark/products/201838/intel-core-i9-10980hk-processor-16m-cache-up-to-5-30-ghz.html, Apr 2020.

[5] W. Ahn, C. Jiang, J. Xu, and M. A. Alam, "A new framework of physics-based compact model predicts reliability of self-heated modern ics: Finfet, nwfet, nshfet comparison," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 13.6.1–13.6.4.

[6] M. A. Anders, H. Kaul, S. Kim, G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag, M. Kar, S. K. Hsu, A. Agarwal, V. Suresh, S. K. Mathew, R. K. Krishnamurthy, and V. De, "25.9 reconfigurable transient current-mode global interconnect circuits in 10nm cmos for high-performance processors with wide voltage-frequency operating range," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 396–398.

[7] C. Auth, A. Aliyarukunju, M. Asoro, D. Bergstrom, V. Bhagwat, J. Birdsall, N. Bisnik, M. Buehler, V. Chikarmane, G. Ding, Q. Fu, H. Gomez, W. Han, D. Hanken, M. Haran, M. Hattendorf, R. Heussner, H. Hiramatsu, B. Ho, S. Jaloviar, I. Jin, S. Joshi, S. Kirby, S. Kosaraju, H. Kothari, G. Leatherman, K. Lee, J. Leib, A. Madhavan, K. Marla, H. Meyer, T. Mule, C. Parker, S. Parthasarathy, C. Pelto, L. Pipes, I. Post, M. Prince, A. Rahman, S. Rajamani, A. Saha, J. D. Santos, M. Sharma, V. Sharma, J. Shin, P. Sinha, P. Smith, M. Sprinkle, A. S. Amour, C. Staus, R. Suri, D. Towner, A. Tripathi, A. Tura, C. Ward, and A. Yeoh, "A 10nm high performance and low-power cmos technology featuring 3rd generation finfet transistors, self-aligned quad patterning, contact over active gate and cobalt local interconnects," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 29.1.1–29.1.4.

[8] C. Bachmann and A. Bar-Cohen, "Hotspot remediation with anisotropic thermal interface materials," in *2008 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, May 2008, pp. 238–247.

[9] A. Bar-Cohen, M. Arik, and M. Ohadi, "Direct Liquid Cooling of High Flux Micro and Nano Electronic Components," *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1549–1570, Aug. 2006.

[10] R. Brain, "Interconnect scaling: Challenges and opportunities," in *2016 IEEE International Electron Devices Meeting (IEDM)*, 2016, pp. 9.3.1–9.3.4.

[11] E. Bury, B. Kaczer, D. Linten, L. Witters, H. Mertens, N. Waldron, X. Zhou, N. Collaert, N. Horiguchi, A. Spessot, and G. Groeseneken, "Self-heating in finfet and gaa-nw using si, ge and iii/v channels," in *2016 IEEE International Electron Devices Meeting (IEDM)*, 2016, pp. 15.6.1–15.6.4.

[12] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov. 2011, pp. 52:1–52:12.

[13] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[14] R. H. Dennard, F. H. Gaensslen, H. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.

[15] Y. Han, I. Koren, and C. A. Moritz, "Temperature Aware Floorplanning."

[16] B. He, M. Wei, S. Somasundaram, C. S. Tan, and E. N. Wang, "Experiments on the ultrathin silicon vapor chamber for enhanced heat transfer performance," in *2016 15th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, May 2016, pp. 569–573, iSSN: 1087-9870.

[17] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, p. 1–17, Sep. 2006. [Online]. Available: https://doi.org/10.1145/1186736.1186737

[18] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, 2001.

[19] M. Huang, J. Renau, Seung-Moon Yoo, and J. Torrellas, "A framework for dynamic energy efficiency and temperature management," in *Proceedings 33rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-33 2000*, Dec. 2000, pp. 202–213.

[20] D. Jang, E. Bury, R. Ritzenthaler, M. G. Bardon, T. Chiarella, K. Miyaguchi, P. Raghavan, A. Mocuta, G. Groeseneken, A. Mercha, D. Verkest, and A. Thean, "Self-heating on bulk finfet from 14nm down to 7nm node," in *2015 IEEE International Electron Devices Meeting (IEDM)*, 2015, pp. 11.6.1–11.6.4.

[21] S. Kondguli and M. Huang, "A case for a more effective, power-efficient turbo boosting," *ACM Trans. Archit. Code Optim.*, vol. 15, no. 1, Mar. 2018. [Online]. Available: https://doi.org/10.1145/3170433

[22] P.-S. Lee and S. V. Garimella, "Hot-Spot Thermal Management With Flow Modulation in a Microchannel Heat Sink," in *Heat Transfer, Part A*, vol. 2005. Orlando, Florida, USA: ASME, 2005, pp. 643–647. [Online]. Available: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1582267

[23] Y. J. Lee, P. S. Lee, and S. K. Chou, "Hotspot Mitigating With Obliquely Finned Microchannel Heat Sink—An Experimental Study," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 3, no. 8, pp. 1332–1341, Aug. 2013.

[24] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2009, pp. 469–480.

[25] Lipeng Cao, J. P. Krusius, M. A. Korhonen, and T. S. Fisher, "Transient thermal management of portable electronics using heat storage and dynamic power dissipation control," *IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part A*, vol. 21, no. 1, pp. 113–123, Mar. 1998.

[26] D. Lo and C. Kozyrakis, "Dynamic management of turbomode in modern multi-core chips," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, 2014, pp. 603–613.

[27] M. Mochizuki, T. Nguyen, K. Mashiko, Y. Saito, X. P. Wu, T. Nguyen, and V. Wuttijumnong, "Thermal management in high performance computers by use of heat Pipes and vapor chambers, and the challenges of global warming and environment," in *2009 4th International Microsystems, Packaging, Assembly and Circuits Technology Conference*, Oct. 2009, pp. 191–194, iSSN: 2150-5942.

[28] R. Mukherjee, S. O. Memik, and G. Memik, "Temperature-aware resource allocation and binding in high-level synthesis," in *Proceedings. 42nd Design Automation Conference, 2005.*, Jun. 2005, pp. 196–201.

[29] S. Mukhopadhyay, A. Kundu, Y. W. Lee, H. D. Hsieh, D. S. Huang, J. J. Horng, T. H. Chen, J. H. Lee, Y. S. Tsai, C. K. Lin, R. Lu, and J. He, "An unique methodology to estimate the thermal time constant and dynamic self heating impact for accurate reliability evaluation in advanced finfet technologies," in *2018 IEEE International Electron Devices Meeting (IEDM)*, 2018, pp. 17.4.1–17.4.4.

[30] A. Pathania and J. Henkel, "Hotsniper: Sniper-based toolchain for many-core thermal simulations in open systems," *IEEE Embedded Systems Letters*, vol. 11, no. 2, pp. 54–57, 2019.

[31] E. Pop, R. Dutton, and K. Goodson, "Thermal analysis of ultra-thin body device scaling [soi and finfet devices]," in *IEEE International Electron Devices Meeting 2003*, 2003, pp. 36.6.1–36.6.4.

[32] Y. Qu, X. Lin, J. Li, R. Cheng, X. Yu, Z. Zheng, J. Lu, B. Chen, and Y. Zhao, "Ultra fast (¡1 ns) electrical characterization of self-heating effect and its impact on hot carrier injection in 14nm finfets," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 39.2.1–39.2.4.

[33] B. C. Schafer and T. Kim, "Hotspots Elimination and Temperature Flattening in VLSI Circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 11, pp. 1475–1487, Nov. 2008.

[34] ——, "Thermal-Aware Instruction Assignment for VLIW Processors," p. 7.

[35] A. J. Scholten, G. D. J. Smit, R. M. T. Pijper, L. F. Tiemeijer, H. P. Tuinhout, J. . P. J. van der Steen, A. Mercha, M. Braccioli, and D. B. M. Klaassen, "Experimental assessment of self-heating in soi finfets," in *2009 IEEE International Electron Devices Meeting (IEDM)*, 2009, pp. 1–4.

[36] B. C. Schäfer and T. Kim, "Autonomous temperature control technique in VLSI circuits through logic replication," *IET Computers & Digital Techniques*, vol. 3, pp. 62–71, 2009.

[37] G. G. Shahidi, "Chip power scaling in recent cmos technology nodes," *IEEE Access*, vol. 7, pp. 851–856, 2019.

[38] S. H. Shin, S. . Kim, S. Kim, H. Wu, P. D. Ye, and M. A. Alam, "Substrate and layout engineering to suppress self-heating in floating body transistors," in *2016 IEEE International Electron Devices Meeting (IEDM)*, 2016, pp. 15.7.1–15.7.4.

[39] K. Skadron, M. R. Stan, W. Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *30th Annual International Symposium on Computer Architecture, 2003. Proceedings.*, June 2003, pp. 2–13.

[40] A. Sridhar, A. Vincenzi, D. Atienza, and T. Brunschwiler, "3d-ice: A compact thermal model for early-stage design of liquid-cooled ics," *IEEE Transactions on Computers*, vol. 63, no. 10, pp. 2576–2589, 2014.

[41] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3d-ice: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2010, pp. 463–470.

[42] M. Stan, R. Zhang, and K. Skadron, "Hotspot 6.0: Validation, acceleration and extension," 2015.

[43] C.-H. Tsai and S.-M. S. Kang, "Standard Cell Placement for Even On-chip Thermal Distribution," in *Proceedings of the 1999 International Symposium on Physical Design*, ser. ISPD '99. New York, NY, USA: ACM, 1999, pp. 179–184, event-place: Monterey, California, USA. [Online]. Available: http://doi.acm.org/10.1145/299996.300067

[44] K. Vaidyanathan, D. H. Morris, U. E. Avci, I. S. Bhati, L. Subramanian, J. Gaur, H. Liu, S. Subramoney, T. Karnik, H. Wang, and I. A. Young, "Overcoming interconnect scaling challenges using novel process and design solutions to improve both high-speed and low-power computing modes," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 20.1.1–20.1.4.

[45] J.-C. Wang and T.-C. Chen, "Vapor chamber in high performance server," in *2009 4th International Microsystems, Packaging, Assembly and Circuits Technology Conference*, Oct. 2009, pp. 364–367, iSSN: 2150-5942.

[46] P. Wang, A. Bar-Cohen, B. Yang, G. L. Solbrekken, Y. Zhang, and A. Shakouri, "Thermoelectric Micro-Cooler for Hot-Spot Thermal Management," in *Advances in Electronic Packaging, Parts A, B, and C*. San Francisco, California, USA: ASME, 2005, pp. 2161–2171. [Online]. Available: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1577475

[47] Q. Wu, M. Martonosi, D. W. Clark, V. J. Reddi, D. Connors, Y. Wu, J. Lee, and D. Brooks, "A dynamic compilation framework for controlling microprocessor energy and performance," in *Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 38. USA: IEEE Computer Society, 2005, p. 271–282. [Online]. Available: https://doi.org/10.1109/MICRO.2005.7

[48] S. Xi, H. Jacobson, P. Bose, G.-Y. Wei, and D. Brooks, "Quantifying sources of error in mcpat and potential impacts on architectural studies," in *International Symposium on High Performance Computer Architecture (HPCA)*, 2015. [Online]. Available: http://www.samxi.org/papers/xi_hpca2015.pdf

[49] G. Yeap, S. S. Lin, Y. M. Chen, H. L. Shang, P. W. Wang, H. C. Lin, Y. C. Peng, J. Y. Sheu, M. Wang, X. Chen, B. R. Yang, C. P. Lin, F. C. Yang, Y. K. Leung, D. W. Lin, C. P. Chen, K. F. Yu, D. H. Chen, C. Y. Chang, H. K. Chen, P. Hung, C. S. Hou, Y. K. Cheng, J. Chang, L. Yuan, C. K. Lin, C. C. Chen, Y. C. Yeo, M. H. Tsai, H. T. Lin, C. O. Chui, K. B. Huang, W. Chang, H. J. Lin, K. W. Chen, R. Chen, S. H. Sun, Q. Fu, H. T. Yang, H. T. Chiang, C. C. Yeh, T. L. Lee, C. H. Wang, S. L. Shue, C. W. Wu, R. Lu, W. R. Lin, J. Wu, F. Lai, Y. H. Wu, B. Z. Tien, Y. C. Huang, L. C. Lu, J. He, Y. Ku, J. Lin, M. Cao, T. S. Chang, and S. M. Jang, "5nm cmos production technology platform featuring full-fledged euv, and high mobility channel finfets with densest 0.021μm2 sram cells for mobile soc and high performance computing applications," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 36.7.1–36.7.4.

[50] S. Yu, H. Yang, R. Wang, Z. Luan, and D. Qian, "Evaluating architecture impact on system energy efficiency," *PLOS ONE*, vol. 12, p. e0188428, 11 2017.

APPENDIX

A. Abstract

HotGauge is a publicly available framework designed to help characterize and evaluate hotspots on modern ICs. It uses modified versions of Sniper for performance simulation, MCPAT for power modeling, and 3D-ICE for thermal simulations.

HotGauge also includes contains a python package that is capable of running thermal simulations using 3D-ICE as well as post processing the thermal data output. This processing includes, among other things, computing the novel metrics developed in this work, namely *MLTD* and *hotspot-severity*, and scripts that plot and visualize the output of the tool. The remaining scripts, from floorplanning to other configuration, are also made to be flexible and support evaluation of other hotspot mitigation techniques as done in the two case studies in this work.

Initially, *HotGauge* was developed on a Red Hat Enterprise Linux 6 server, but it has also been validated on Ubuntu 20.04. Included in this software release is a *Dockerfile* that allows evaluation and use of *HotGauge* on a custom Docker container based on Ubunutu 20.04.

B. Artifact check-list (meta-information)

- **Program: HotGauge**
- **How much disk space required (approximately)?: 100MB for base download, plus file size of 3D-ICE, Sniper, and MCPAT**
- **How much time is needed to prepare workflow (approximately)?: 1 hour**
- **Publicly available?: Yes**
- **Code license : BSD 3-clause clear**
- **Archived: Zenodo DOI 5523535**

- **Most Recent Archive : Zenodo DOI 5523504**

C. Description

1) How to access: The *HotGauge* framework is available on Zenodo and GitHub.

2) Software dependencies: HotGauge was developed on a RHEL 6 server and has also been validated on Ubuntu 20.04. The provided *Dockerfile* will create a Docker container based on Ubuntu 20.04 that will run *HotGauge*. Any system with Docker installed and configured should be able to run the Docker container.

D. Installation

After downloading the repository, *HotGauge* can be leveraged inside of a Docker container by first building the container and running it using the provided shell scripts, called `./docker_build.sh` and `./docker_run.sh`, respectively.

Alternatively, *HotGauge* can be set up on the local machine by following the instructions included *README.md* file, which also contain hints to help compile the tools used by *HotGauge*.

E. Experiment workflow

To set up *HotGauge*, simply activate the ptyhon virtual environment that was created when initially setting up *HotGauge*. In the Docker container, this is done via the following:

```
cd ~/HotGauge
source env/bin/activate
```

Once the virtual environment is active, run any of the scripts inside the *examples/* directory. For example, `python simulation_with_warmup.py`. Once this is complete, you can try out the example post-processing scripts inside the *simulation_with_warmup/* directory. See the *README.md* file in that directory and comments inside the shell script files for details.

F. Evaluation and expected results

Running the above example script will produce outputs in the *HotGauge/examples/simulation_with_warmup/outputs/* directory. Within that directory are the results of the warmup simulation (*outputs/warmup*) and the thermal simulation of the example workload (*outputs/sim*). The post-processing scripts produce outputs in *outputs/sim* and figures in *outputs/sim/plots/*.

Output files can be copied out of the container by copying to the docker volume inside the container located at */data/*. This volume can be located on the host machine via `docker volume ls` and then running `docker inspect <container-id>` with the container ID of the appropriate docker image. Note, this volume persists on the host machine but not across calls to `docker run`.

G. Experiment customization

While scripts can be customized and written in the *HotGauge* Docker container, it is recommended that *HotGauge* is installed locally so as to avoid issues of having to copy files to and from the container.