HotSpot: a Dynamic Compact Thermal Model at the Processor-Architecture Level

Mircea R. Stan [a], Kevin Skadron [b], Marco Barcella [a], Wei Huang [a], Karthik Sankaranarayanan [b], and Sivakumar Velusamy [b]

[a] *Dept. of Electrical and Computer Engineering*

[b] *Dept. of Computer Science*

University of Virginia

Charlottesville, VA 22904

Accepted for the Special issue on Therminic 2002 as paper T02MEJ_14

Abstract

This paper describes a thermal-modeling approach that is easy to use and computationally efficient for modeling thermal effects and thermal-management techniques at the processor architecture level. Our approach is based on modeling thermal behavior of the microprocessor die and its package as a circuit of thermal resistances and capacitances that correspond to functional blocks at the architecture level. This yields a simple compact model, yet heat dissipation within all major functional blocks and the heat flow among blocks and through the package are accounted for. The model is parameterized, boundary- and initial-conditions independent, and is derived by a structure assembly approach. The architecture community has demonstrated growing interest in thermal management, but currently lacks a way to model on-chip temperatures in a tractable way. Our model can be used for initial exploration of the design space at the architecture level. The model can easily be integrated into popular power/performance simulators, can be used to determine how thermal stress is correlated to the architecture, and how architecture-level design decisions influence thermal behavior and related effects.

Key words: Thermal models, Dynamic compact models, Computer Architecture

1 Introduction

Many analysts suggest that increasing power density and resulting difficulties in managing on-chip temperatures are some of the most urgent obstacles to continued scaling of VLSI systems within the next five to ten years. Just as has been

done before for power-aware design, "temperature-aware" computing must be approached not just from the packaging community, but also the VLSI and processor-architecture communities. In particular, processor-architecture solutions can often use global, runtime knowledge to change the behavior of large portions of the processor (typically with the support of appropriate circuit techniques). There is growing interest in architecture-level solutions, as evidenced by recent work in this field [1–4]. Yet the architecture community is currently completely lacking a way to model temperature at any level of granularity other than low-level circuits!

Computer-architecture studies have unique requirements that preclude the use of traditional thermal-modeling techniques like [5–11]. Such previously proposed techniques usually assume an existing circuit (or at least a low-level RTL, VHDL, or Verilog description) and conduct joint electro-thermal simulation, often including the role of the thermal package. In contrast, the kind of architecture studies we are targeting are broad power/performance design-space studies to choose the overall pipeline organization, structure capacities (e.g., cache sizes) and algorithms (e.g., branch-prediction algorithms). These simulations typically use cycle-by-cycle behavioral models with high-level analytic power models at the granularity of architectural units, simulate large benchmarks for hundreds of millions to billions of clock cycles, and may be conducted long before any detailed design has begun, let alone prototype chips and packages are available. This high-level, early-stage modeling precludes thermal-response measurements, e.g. [10,11], and the use of analytic power models obviates the need for joint electro-thermal modeling [6–10]. Even the logi-thermal simulation described in [10] is too detailed to be compatible with early architectural simulation environments. Yet the microarchitecture's ability to control entire regions of the chip makes architectural thermal control a viable option in conjunction with circuit and packaging techniques.

For example, processor manufacturers have begun to deploy thermal packages designed for less than the worst-case power dissipation, with on-chip temperature sensors and autonomous runtime response (often called "dynamic thermal management" or DTM) by the chip to throttle down power dissipation if on-chip activity exceeds the package's capabilities and temperatures rise beyond safe limits. The Intel Pentium 4, for example, uses a package designed for 80% of worst-case power dissipation and relies on clock gating when temperatures exceed safe limits [12]. Accurate placement of sensors is challenging in its own right, but assuming that the sensor(s) will detect unsafe temperatures, *performance* becomes the architectural design challenge. Higher-performance DTM solutions ensure that unusually hot applications will not suffer too much, and the higher the DTM performance, the lower it is possible to push the cost of the package. Our prior work [4] showed that open-loop techniques like clock gating and fetch toggling [1] fail to account for program-specific behavior and instruction-level parallelism, and impose unnecessarily high performance penalties. Instead, we find that feedback control can tune the runtime response to the degree of thermal stress. At an even higher level of abstraction, operating-system scheduling decisions can be made according

to current operating temperatures and observations of each thread's thermal behavior [3]. Ultimately, these operating-system and micro-architecture techniques will be combined, especially in chip-multiprocessors where "core-hopping" can be used to match workloads to per-chip and per-structure temperatures.

One obstacle to this kind of work is the lack of thermal simulators that are amenable to use in high-level architecture research environments. Yet the accuracy of thermal modeling has a substantial effect on the accuracy of thermal-management studies at the processor-architecture level and the conclusions they draw [4]. Without this essential modeling capability, architecture researchers are therefore limited to crude and inaccurate estimation techniques and are unable to effectively develop and evaluate techniques for thermal management. For example, we show that simplistic thermal estimation based on simple averages of runtime power density using architecture-level analytical power models like [13] yields poor accuracy and may even target the wrong architectural structures.

A thermal model that processor architects can use to evaluate thermal behavior of various designs, or to develop runtime architecture techniques for thermal response, therefore necessitates a parameterized, "a priori" thermal modeling approach that can directly model temperature using only a high-level model of the processor architecture and that can be automatically configured to simulate a wide range of architectures. The model should therefore be based purely on geometric considerations and material properties, and the dynamic simulation should only depend on dynamic information about power density in each architectural unit.

This paper describes *HotSpot*, a thermal modeling framework for processor architects that meets these goals and can be incorporated into popular cycle-accurate processor-architecture power/performance simulators like Wattch [13]. HotSpot consists of four major parts:

(1) *HotArea*: A module that estimates on-chip areas for the major blocks of a specified microprocessor configuration.
(2) *HotFloorplan*: A module that uses the specified processor configuration and the corresponding area estimates to derive a high-level floorplan.
(3) *HotPackage*: A module that estimates an equivalent thermal resistance and capacitance for a specified package configuration.
(4) *HotBlocks*: A module that derives—using input from the above three steps—an equivalent dynamic compact model to approximate heat flow in the microprocessor and its package. The model can be characterized as a parameterized, boundary and initial-conditions independent (BICI) dynamic compact thermal model derived with a structure assembly approach [14]. The model is simple and therefore computationally efficient to recompute after every clock cycle in a cycle-level power-performance simulator.

These are integrated into a portable software package that can easily be incorpo-

rated into architecture-level simulators.

This work focuses on modeling heat dissipation and heat flow—that is, actual on-chip temperatures. This can be used to determine how thermal stress is correlated to the architecture, and how architectural design decisions influence thermal behavior and related effects like reliability and leakage currents (leakage-control techniques have become a major area of investigation in processor architecture, despite the lack of temperature-aware models). Clearly this approach entails a number of approximations and is not meant to replace other forms of thermal simulation for circuit or package design. Rather, HotSpot allows early-stage architecture explorations that facilitate architecture-level thermal solutions that are synergistic with thermal solutions at other levels of abstraction.

This paper reports our ongoing work to develop and refine HotSpot. The next section gives an overview of our approach to modeling dynamic thermal behavior from the architecture standpoint, and subsequent sections describe the four components of the HotSpot framework. We describe the HotArea and HotPackage components briefly; the main focus of this paper is a description and demonstration of the HotFloorplan and HotBlocks techniques. Finally, we conclude the paper and discuss areas for future work.

2 Thermal Modeling at the Processor Architecture Level

For the kinds of studies we intend to pursue, the compact model must have the following properties:

- It must track temperatures at the granularity of individual pipeline units. This means the equivalent RC circuit must have at least one node for each unit.
- It must be computationally efficient to solve the RC circuit's differential equations, so that this task does not substantially slow down power/performance simulations.
- It must be parameterized, in the sense that a new compact model is automatically generated for different microarchitectures.
- It must be portable, making it easy to use with a range of power/performance simulators.
- It must be boundary- and initial-condition independent (BICI) [14]. The thermal model component values should not depend on initial temperatures or the particular configuration being studied.
- It must be calibrated so that simulated temperatures can be expected to correspond to what would be observed in real hardware.

The model we have developed, which we call HotSpot, meets all these conditions at the expense of a reduced accuracy compared to more detailed approaches. For

an integrated circuit at the die level, heat conduction to the package and heatsink, and convection from the heat sink to ambient are the dominant mechanisms that determine temperatures on the die. There exists a well-known duality between heat transfer and electrical phenomena [15]. Heat flow can be described as a current, and the passing of this heat flow through a thermal resistance leads to a temperature difference equivalent to a voltage. Thermal resistances are enough for describing steady-state behavior, but dynamic behavior is also important for dynamic thermal management, and this requires thermal capacitances as well. Because of thermal capacitances, even when the power flow changes instantaneously, there is a delay before the temperature changes and reaches steady state. The thermal resistances and capacitances together lead to rise and fall times characterized by thermal time constants exactly analogous to the electrical time constants for an RC circuit. Prior work on thermal issues in the architecture field has not modeled temperature at all, but rather used a mere average of power dissipation values as a proxy. HotSpot provides a direct model of temperature. It uses the duality between electrical and thermal resistances and capacitances to derive a dynamic compact model of the heat flow among the different architecture-level blocks within a microprocessor. Architectural power-performance simulators like Wattch [13] can then be used to determine cycle-by-cycle power dissipations and feed them into the circuit model to compute cycle-by-cycle heat flow.

Large ICs have a heterogeneous structure with many different areas on the die working at different activity factors, which implies that power is not dissipated uniformly on the chip. This *spatial non-uniformity* is complemented by a *temporal non-uniformity* as many structures on the die go from idle mode to full active mode, and vice-versa, at different times. Recent emphasis on low-power design techniques such as power-down modes, clock domains and clock gating, etc. are only exacerbating the spatial and temporal non-uniformity for on-chip power density. As a result of this non-uniformity, the chip will exhibit so-called “hot spots”. These hot spots have a spatial distribution as a result of the non-zero thermal resistivity of silicon and also a temporal distribution due to changing program behavior and the time constants implied by the thermal mass (thermal capacitance).

In order to derive a compact model one needs to decide on the level of granularity for the lumped elements. For pipeline-level modeling, we use a natural partitioning where functional blocks on the chip are the nodes in the lumped circuit model. Thus there is a one-to-one correspondence between the model in a power/performance simulator like Wattch and the model in the thermal simulator, which leads to a straightforward coupling between the two.

The equivalent-circuit model is dependent on the floorplan and packaging of the processor being simulated, while the floorplan and the calculation of values for thermal resistance and capacitance are dependent on the areas of the functional blocks of interest.

Although beyond the scope of this work, a complete modeling and evaluation tool for thermal management must account for the behavior of temperature *sensors* as well as the actual on-chip temperatures. Our understanding is that the accuracy of most practical sensor designs is susceptible to both process variations during manufacturing and power-supply variations. Compensating for these requires increased transistor sizes in the sensors, with corresponding increases in power. In current microprocessors, this leads to the use of only a small number of sensors, e.g. four sensors distributed around the die. Because these sensors may be remote from the hot spots of interest, there may be a time lag between localized heating that should generate a thermal-management response and the actual observation of any change in temperature at the sensor. An architecture-level model of these effects will eventually be incorporated into HotSpot.

We reiterate that our approach is motivated by a microarchitecture-centric viewpoint. This means that we neglect a number of issues that are typically considered in a full thermal design. For example, we completely neglect thermal modeling of the circuit board, heating of the air within the computer system case, temperature non-uniformities in the air, heat flow through the package pins, etc. Because microarchitecture simulations only model very small time periods of perhaps a few seconds at most, these components that are external to the chip and its package respond too slowly to change temperature during such short time scales, and these external components do not contribute in a significant way to the capacitance being driven by the chip. Indeed, even the heat sink itself does not change temperature on these time scales, but it must be included because its capacitance does affect the transient behavior of the chip. The intended purpose of HotSpot for use with microarchitecture simulators also drives some of the approximations and simplifications that we make. The purpose of HotSpot at this point in time is therefore not to perform reliability analysis or package design, but rather to allow research into microarchitecture techniques for runtime mitigation of hotspots that develop according to the behavior of specific architectural structures, like register file or cache activity.

3 HotArea: Estimating Area for Microprocessor Functional Units

Efficient architecture-level thermal modeling requires estimation of the area of various functional units on chip based on architectural parameters. The area estimation itself can also allow the designers to make more comprehensive design space comparisons in the early phase of the design process. The area-estimation tool is still in its early stages, and the HotFloorplan and HotBlocks modules can instead use published area information for the time being.

In this section we give a brief overview of our strategy for area estimation. We divide the blocks that constitute a microprocessor into three categories, accord-

ing to their regularity. The first class consists of structures with two-dimensional regularity, generally memory structures. These include caches, TLBs, register file, branch predictor, load-store queue, instruction-issue queues, and active list. The second class consists of structures with one-dimensional regularity, generally bit-slice structures. These include all integer datapath and floating point units. The third class consists of irregular structures, generally control logic. These include the memory-management unit, decoding logic, register-renaming logic, issue logic, and graduation logic. We use analytic models to derive a "baseline" area for each block from a known chip layout, and then use known measurements from the known layout to scale this estimate. This scaling factor can then be applied for using the analytic models to derive area estimates for different architectures. For example, once the scaling factors for first-level caches, ALUs, etc. are known, different architectures can be explored with larger/smaller caches, different numbers or types of ALUs, etc.

We use an existing tool, Cacti 3.0 [16], to calculate the area of memory structures; Cacti is in turn based on [17]. Area estimation for the register file, queue, and cache structures can be implemented by simply inputting the proper parameters to Cacti.

Since arithmetic units can be implemented in many different ways, we consider area estimators for the most commonly used structures. For example, in providing area models for adders, we model both a Kogge-Stone and a Han-Carlson structure [18], letting the users choose between them. The datapath is normally on the critical path of the processor, so it requires high performance. We accept as a parameter the kind of CMOS circuit (dynamic/static) used to implement it and allow the user to specify a sizing parameter, since the execution units are most likely sized larger than minimum size.

For irregular structures, we adopt the model proposed by Nemani [19]. First we need an approximate boolean description of the structures, which is plausible to derive for decoding and at least the major aspects of memory-management, register-renaming, issue, and graduation. Then we use the model to estimate the transistor number. Finally we use average transistor density to estimate the area.

Cacti is an existing tool that has already evolved through three major releases. Our area estimation for bit-sliced structures is producing early estimates within 50% of areas suggested by published die photos and floorplans. The major remaining work is the estimation of irregular structures.

After the HotArea tool completes its analysis, it supplies <area,aspect ratio> tuples, which can be provided to the HotFloorplan and HotBlocks components of the framework.

4 HotFloorplan: Modeling Thermal Adjacency Using Floorplanning

Current microprocessor simulators at the architecture level do not model floorplan information. However, with the growing importance of temperature and power effects, as well as the disproportionate scaling of logic and wire delays, knowledge of the layout of the functional units is becoming essential for architecture-level modeling of performance, power, and heat. Modeling these issues effectively at the architecture level necessitates a simple floorplanning tool that does not require lower-level behavioral or synthesis information, because architecture research often wishes to study tradeoffs related to performance, power, or heat for novel architectures that may never be realized in lower-level descriptions that traditional floorplanning algorithms require. The HotFloorplan tool incorporates a new algorithm that is suited for architecture-level planning and research, and provides the architecture researcher an easy-to-use and generic interface for incorporating floorplanning information into temperature, power, and performance models.

A typical ordering of pipeline stages in an out-of-order superscalar microprocessor is as follows: instruction fetch, decode, rename, dispatch, issue, execute and commit. The corresponding functional blocks involved in this sequence are instruction cache (I-cache), instruction TLB (I-TLB), and branch predictor; decoder; renamer; issue queues; execution units and data cache (D-cache); and reorder buffer (some blocks like the register file, the reorder buffer, etc. are accessed in multiple stages of the pipeline; also, some pipeline stages access more than one functional block.) We have observed that published floorplans of processor chips reflect this pipeline structure. Units involved in adjacent pipeline stages are typically also adjacent in the floorplan. Thus the chip-level physical adjacency is captured well by the "functional adjacency" of the blocks in the pipeline. This suggests that the ordering of the pipeline stages is a good way to express floorplans at the architectural level. This is the approach we take in the HotFloorplan tool. Traditional floorplanning algorithms, e.g. simulated annealing [20] are not appropriate here since they require "quantitative" information about the connectivity between different blocks which is available at the computer architecture level only in a "qualitative" sense.

The user pictures the floorplan as a set of concentric semicircular "strips". Each strip corresponds to a sequence of functional blocks. To express a floorplan, the user lists the functional blocks in each of these strips clockwise along with their areas (which are taken from the HotArea tool). The clockwise ordering of functional blocks in each of these strips will be very similar to the ordering of pipeline stages of the processor. The actual floorplan is generated from this specification. The semicircular strips are fit into a bounding square (the chip) by adjusting their widths. The assumption is that the functional blocks are "soft" blocks, so their aspect ratios can change to accommodate this fitting. Apart from the ease of specification, another advantage of these semicircular strips is the ease of modification. Changes to an existing floorplan can be made very easily. For example, increasing

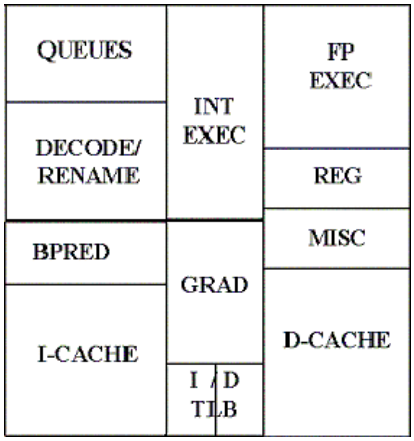


Fig. 1. The semicircular floorplan generated by the HotFloorplan tool using the MIPS R10000 configuration.

the area of a few blocks simply causes the tool to widen the strips, while changing the adjacency involves just changing the order of blocks within a strip.

In addition to the above form of expressing and computing floorplans, future work will allow the user to completely specify a floorplan using vertical and horizontal adjacency matrices. This is desirable when simulating a system with a known floorplan. Moreover, a query library will be provided which can be used to get more information about the floorplan, like adjacencies, distances between blocks, etc.

Input to the HotFloorplan tool can come either from the HotArea tool or from published or other externally derived area estimates. To verify the Floorplan tool, we make some approximate area estimates for each of the functional blocks. This is done by manually estimating areas using the published photographs of various processor dies and taking an average of the block sizes across processors. We then modeled a floorplan that has two strips. The outer strip has the functional unit sequence: I-cache, branch predictor, decode/rename unit, queues, integer execution units, floating point execution units, register file, miscellaneous structures like bus interface, and finally D-cache. The inner strip has the functional unit sequence: I-TLB, graduation unit and D-TLB. This specification is also the default case for our tool: when the user does not specify any floorplan information, this is the configuration assumed. The floorplan generated for this default specification is shown in Figure 1.

In order to compare this default floorplan with the floorplans of contemporary microprocessors, we took the die photographs and floorplans of three processors: the MIPS R10000 [21], the Pentium [22], and the Alpha 21264 [23]. For simplicity, we merged the manufacturer-specified functional blocks in these floorplans so that they approximately match the blocks of the default floorplan. We then flipped andor rotated these floorplans such that the I-Cache and D-Cache were in positions similar to the default floorplan. The resulting floorplans are shown in Figure 2. It can be seen from these figures that the blocks adjacent to each other in the floorplan are

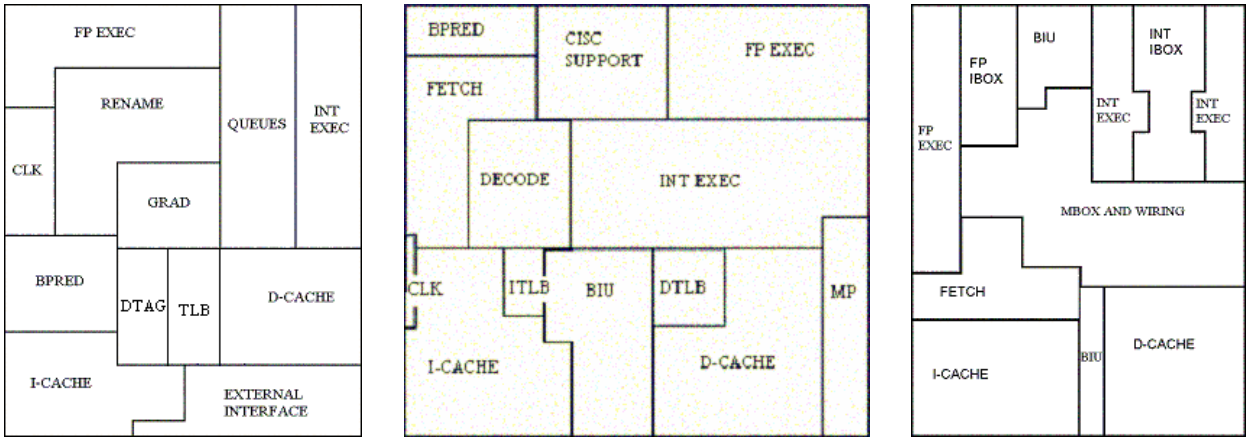


Fig. 2. “Normalized” floorplan for the MIPS R10000 (left), Intel Pentium (middle), and Alpha 21264 (right) processors.

typically the blocks that are adjacent in the ordering of the pipeline stages. Moreover, the default floorplan, which is generated using the ordering of pipeline stages, qualitatively resembles the real floorplans.

In order to quantitatively characterize the similarity between the default floorplan and these real floorplans, we scaled each of the real floorplans to the size of the default floorplan. We then measured the coordinates of the centers of various functional blocks in each of these scaled floorplans. Further, we computed the Euclidean distance between these centers and the centers of the corresponding blocks in the default floorplan. Table 1 below shows these distances normalized to the height of the default floorplan. Note that we did not adjust the default floorplan in any way to account for the different processor floorplans. When trying to replicate a specific floorplan, we have found that we can do so almost exactly.

Table 1
Distances from computed vs. observed locations of the major functional blocks. Results are in percent, normalized to the height of the chip.

	Alpha	MIPS	Pentium
I-cache	6.4%	5.4%	11.1%
D-cache	3.9%	10.8%	9.1%
Integer execution	NA	36.8%	24.8%
FP execution	71.2%	49.1%	5.6%
Queues	NA	50.8%	21.7%
Branch pred.	4.4%	7.9%	52.3%
Decode/rename	37.9%	22.7%	7.1%

In this table, smaller values mean greater similarity with the default floorplan. The error in placement ranges from 4% to 71%; most values are less than 25%. The 71% value is due to the fact that our simple rotation and mirroring approach is not

sufficient. In both the Alpha and our computed floorplan, the floating-point unit is opposite the caches, but on the left-hand side in one case, the right-hand size in the other case. If we mirrored only the top half of the Alpha floorplan, there would be perfect agreement for the FP unit. Of course, all values are approximate, since they result from manual observation of die photographs and the aspect ratios vary. For instance, the R10000 photograph has many empty spaces between functional blocks. In making it comparable in structure to the default floorplan, these empty spaces were merged into the functional blocks. Also, the "integer execution" and "queues" rows of the Alpha processor column are empty because the integer execution units and the queues are split in the floorplan. So we were not able to make a reasonable measure of "center coordinates" for them.

Since the I-cache and D-cache are approximately in the same position across all the floorplans, those two rows have good values in the table. It can also be seen that the 21264 and R10000 branch predictors are in a position similar to the default floorplan (near the I-cache) while the Pentium branch predictor is not. Similarly, it can be seen that the Pentium decoder is in a position similar to the default floorplan while it is not the case with the other two.

Considering the wide variety of floorplans that real chips exhibit—demonstrated well by the three floorplans in Figure 2—it is impossible for a high-level modeling tool to automatically reproduce the exact floorplan of a particular processor without manual intervention. Since a great deal of customization goes into most of these floorplans, it is possible that no traditional floorplanning algorithm could reproduce them either. Our goal is not to develop an algorithm that can accomplish this impossible task, but rather to develop an algorithm that can produce floorplans with reasonable adjacencies for high-level modeling of thermal, power, and delay effects. Furthermore, many architecture simulation studies explore hypothetical processor configurations that have no lower-level specifications that can be used with traditional floorplanning algorithms but still need some reasonable estimate of where various functional units would appear in an eventual floorplan. Both the quantitative results and the visual similarity between the default floorplan in Figure 1 and the real floorplans in Figure 2 suggest that our high-level floorplanning tool does a good job of producing reasonable floorplans and can be used to provide sufficiently realistic adjacency information for high-level performance, power, and thermal studies. It is true that the final floorplan for an actual implementation may differ from the one modeled using HotFloorplan. On the other hand, HotFloorplan can replicate known floorplans almost exactly. In either case, as a design develops, continuing refinement of the floorplan and thermal modeling is needed as a design proceeds through various stages of implementation to ensure that design decisions remain valid. Nevertheless, HotFloorplan allows the study of layout-dependent design choices to begin much earlier and allows new hypothetical architectures to be investigated in ways that were not previously possible.

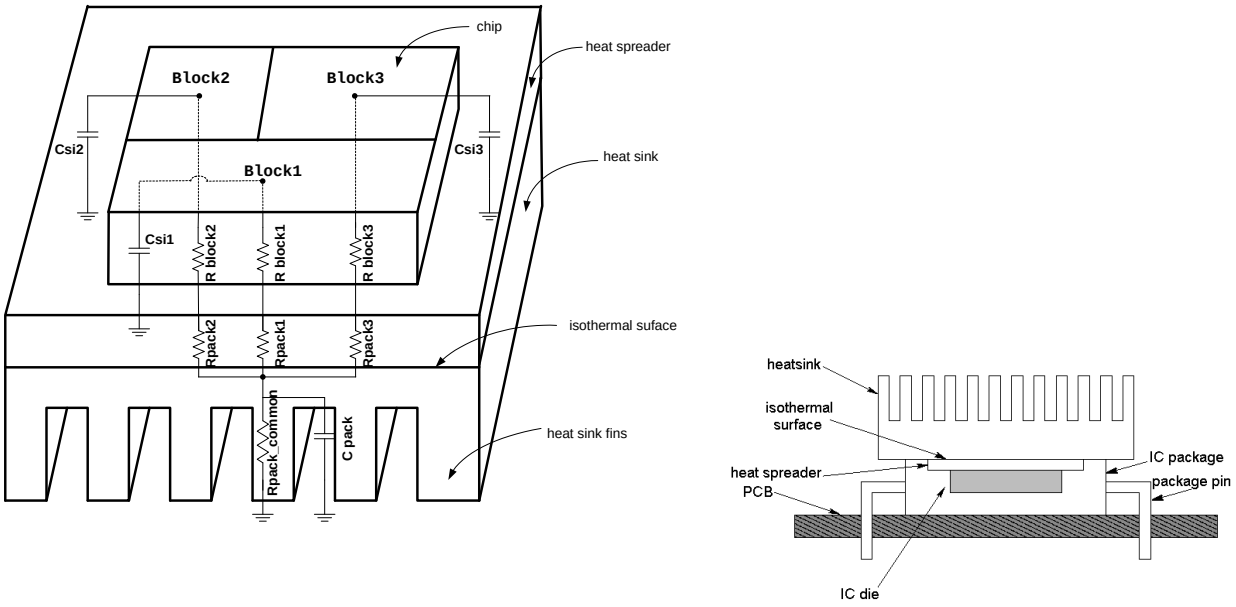
5 HotBlocks: a Dynamic Compact Thermal Model

The HotBlocks model is parameterized, boundary- and initial-conditions independent (BICI), and is derived with a structure assembly approach based on physical properties of materials and the dimensions of the various structures [14]. The equivalent dynamic compact model is generated automatically, and, supplied with power dissipations over any chosen time step, it computes temperatures at the center of each block and also at the center of each shared edge between two blocks.

The equivalent circuit consists of two major components, a "vertical" model to capture heat flow into and within the thermal package from the driving point thermal impedances, and a "lateral" model to capture lateral heat flow among pipeline units via transfer impedances. We explain these two aspects of our model in turn. We point out here that we have made a number of simplifications and approximations in the model that we believe make it more accessible and convenient to use for architecture studies, at the cost of making the derivation more empirical and ad-hoc in nature. For example, collapsing the different layers (see below) corresponds to the fact that architecture studies rarely have detailed geometric or material information about the thermal package, since that has yet to be chosen. For the types of transient effects we are studying, we believe these approximations do not improperly sacrifice relative accuracy in terms of the thermal behavior between different architecture techniques.

The vertical model, shown in Figure 3a, accounts for heat flow from each block in the die into the package, as well as heating of other blocks due to heating of the package. From detailed finite-element method simulations we observed that while at the surface of the die there are significant gradients, as the heat flows to the ambient the magnitude of the gradients in the heatsink decreases to the point where there exists a virtual isotherm at some depth within the heatsink. The vertical model captures the separate heat flow from each block to the isotherm, followed by combined heat flow from the isotherm to the ambient. Note that, to more clearly show the RC network we are describing, in Figures 3a and 4b the chip and package are shown upside down relative to typical use as in 3b.

The first element of the vertical model consists of an RC pair from the center of each unit to the thermal package; this accounts for the driving point (heating in the silicon due to the power being generated there) and how that heat flows into the package. The RC values are computed from the material properties of silicon. The second element is a second set of RC pairs connected in-between the first set and the isotherm. These account for vertical transfer impedances or heat flow in the package. Finally, we have a single RC pair from the isotherm for the package and the convection to air. This accounts for heat flow into the fins of the heat sink and also heat transfer to the air. Air is assumed to be at a fixed ambient temperature, which is generally assumed in thermal design to be $45°$C [24] (this is not



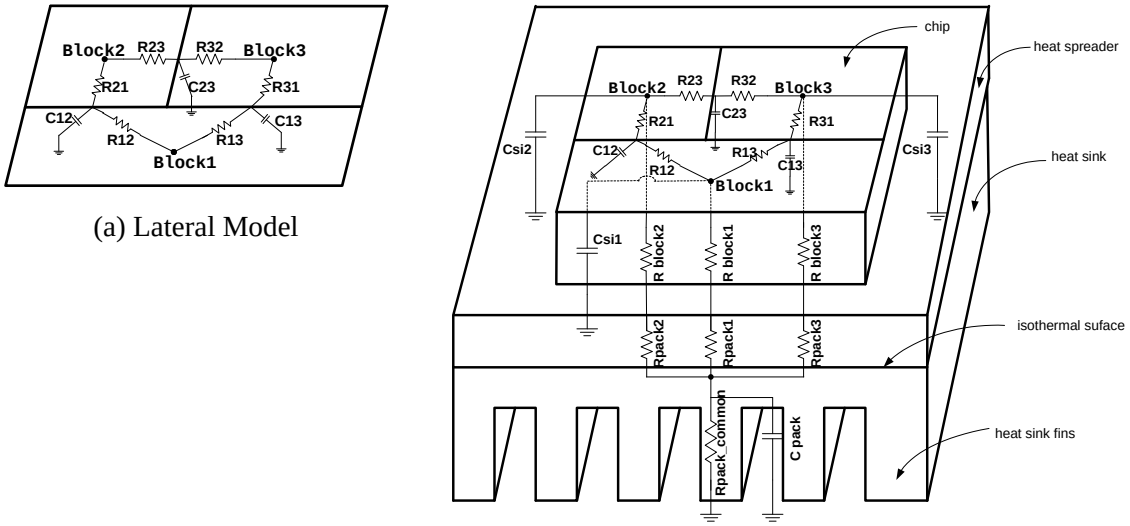
(a) Vertical model (b) Typical Package

Fig. 3. (a): Example vertical RC network, with an RC pair from the center of each unit to the thermal package; an RC pair for each corresponding block of the package; an RC pair for each corresponding block of the heat sink above the isotherm; and a shared RC pair for the thermal package below the isotherm. For clarity, this image is upside-down relative to the typical use as in (b): Side view of a typical package configuration upon which the HotSpot model is based. The model combines the heat spreader and heat sink into a single unit.

the room ambient, but the temperature inside the computer “box”). Although the vertical model may seem complex, the RC pairs for the various layers are in series and do not significantly increase the complexity of the circuit solver. Note that the isothermal surface—the point at which temperature is uniform across the width of the package, does not necessarily lie at the spreader-sink interface, but rather somewhere within the heat sink.

The lateral model, shown in Figure 4a, accounts for lateral transfer impedances, heat that directly flows in the lateral direction. It consists of an RC pair from the center of each unit to the middle of each edge with a shared block. The locations and adjacencies of the units are determined by a prior floorplanning step, which we discuss in Section 4. Here we illustrate the lateral model using a simple floorplan that consists of only three blocks. We only consider lateral heat flow between adjacent functional blocks and ignore any direct thermal impedances between non-adjacent blocks since this simplifies the thermal circuit and leads to negligible differences in final results.

Because the thermal conductivity of the heat spreader and heat sink are much greater than that of silicon, substantial lateral heat flow occurs through the package.



(a) Lateral Model

(b) Complete Model

Fig. 4. (a): Lateral model, with an RC pair between the center of each node and the center of each shared edge. (b):Overall RC model, including both vertical and lateral components.

Ideally, the lateral model therefore has three layers: one for lateral heat flow in the silicon; one for lateral heat flow in the spreader (made of aluminum, copper, silicon carbide, or some organic material); and one for lateral heat flow in the heat sink. Between the die and spreader layers, and the spreader and sink layers, would be vertical resistances to model the non-ideal heat conductivity of the interface layers (typically phase-change films, thermal grease, etc.). Since these interface layers are so thin as to have essentially no thermal mass and they are a small constant scaling factor, we account for them in the α scaling factor below.

Such a multi-layer model leads to an RC model that would be too complex to solve at the simulation rate we desire and might require more detailed design information that architects might not possess. Instead, we collapse the three layers into a single lateral layer using a scaling factor derived from our reference model—see below. In this way we accurately capture the lateral heat flow in all three layers based only on architectural, area, and floorplan information and the package information we calculate below. The combined model is shown in Figure 4b.

The resulting model accomplishes the goal of a paramterized model that can be derived in a straightforward fashion from the high-level specifications that architects typically work with, and avoids the need for low-level simulations or direct physical measurements that thermal multi-port techniques typically require.

6 HotPackage: Accounting for Package Design

Packages today typically consist of a spreader plate, often made of aluminum, copper, or some other highly conductive material, a heat sink of aluminum or copper, and a fan. To determine a cost-effective package that is not designed for the worst-case power dissipation, we used our power/performance model to determine the average power dissipation of all the SPEC2000 benchmark programs [25] commonly used in computer-architecture research. We then selected a package that would allow a modest number (six) of benchmarks to exceed 85° (four by a large amount, two by about 1°, and three that are close but do not exceed 85°). The resulting package has a thermal resistance of 1.0 K/W, which includes the spreader, interface layers, heatsink and heatsink to air convection. This is based on published data as that in table 2. In contrast, to design for the the hottest application, *equake*, the package would need a thermal resistance of 0.54 K/W, which is much more expensive.

Table 2
Published and calculated values for thermal resistance for four heat sinks. Published values are taken from http://www.allstarcomponents.com/cooling.html. "Rect" means rectangular.

Material	Area (cm^2)	Base Thick (cm)	# Fins	Fin Type	Fin Ht. (cm)	Air Sp. (m/s)	Spec. R (K/W)	Calc. R (K/W)
Al	36	0.69	28	Rect	2.06	5	0.67	0.613
Cu	61	1.17	256	Pin	1.68	10	0.36	0.38
Al	64	0.60	18	Rect	2.4	8	0.60	0.62

7 HotSpot: Deriving the Lumped Thermal R/C Values

In this section we sketch how the values of R and C are computed. Some parameters need to be empirically determined, which requires a reference model. The most accurate way to validate our model would be to compare temperature predictions from HotSpot with time-series measurements of per-block operating temperatures for a microprocessor running a real workload. Since there exists no infrastructure for making such measurements, we instead created an independent model of this system in Floworks [26], a commercial, finite-element (FEM) simulator of 3D fluid and heat flow for arbitrary geometries, materials, and boundary conditions.

The derivation is chiefly based upon the fact that thermal resistance is proportional to the thickness of the material and inversely proportional to the cross-sectional area across which the heat is being transferred:

$$R = \frac{t}{k \cdot A} \tag{1}$$

where k is the thermal conductivity, $100W/m \cdot K$ for silicon at $85°$C. Thermal capacitance, on the other hand, is proportional to both thickness and area:

$$C = c \cdot t \cdot A \tag{2}$$

where c is the thermal capacitance per unit volume, $1.75 \times 10^6 J/m^3 \cdot K$ for silicon and $3.55 \times 10^6 J/m^3 \cdot K$ for copper. Calibration with our Floworks FEM simulation has shown that a scaling factor is necessary for the capacitance (c) values: 0.47 for silicon and 0.62 for copper. In exploring different configurations, we have come to the conclusion that this factor is partly due to an aspect-ratio effect, because the die especially is extremely thin and does not absorb heat uniformly throughout its volume, and partly a consequence of the difference between lumped versus distributed RC models [18].

For our 1 K/W package, we must deal with the fact that the isotherm is not on the interface between the spreader and sink. The location of this isotherm depends on the thickness of the chip. Using our reference model in Floworks, we simulated a variety of package configurations with various thicknesses for the die and spreader, and various power-density profiles on the chip, in order to find the isothermal surface. We found a consistent linear relationship that the fraction of the package that is not isothermal is given by $s = 120 \cdot t/R_{pack}$, so for a chip thickness $t = 5.0 \times 10^{-4}$m that we use, $s = 0.06$. This means that 0.06 K/W must be shared among all the blocks (inversely proportional to their area), with the remaining 0.94 K/W in a single common R_{pack_comm}.

For the lateral components, Rs and Cs are computed from the center of each block to the center of each block's shared edge. The Rs and Cs must also be scaled to account for the collapsing of the three layers, and to account for the impact of aspect ratio. Several steps are therefore required.

First, for each block the bulk resistance R_{bulk_i} is computed using Equation (1). This is then scaled by $\alpha = 0.018$ to account for the collapsing of the lateral layers. α is determined by the thermal conductivity of the metal materials comprising the spreader and sink, and the much larger thickness of the package compared to the silicon. It is found empirically using Floworks, as described in [14]. The value of α is independent of block size, aspect ratio, chip/package thickness, power density, and technology. Finally, R_{bulk_i} is divided into individual resistances corresponding to the shared edges with neighboring blocks.

This assumes a square block. But the heat flow is also dependent on the aspect ratio; as a block becomes narrower, more heat flows across the long edge than across the short edge, requiring the introduction of a scaling factor β that models this difference. Thus we compute the R in each dimension, x and y, to each of the four directions (up, down, left, right) as:

$$R_x = \left[\beta \cdot (\sqrt{\frac{L}{W}} - 1) + 1\right] \cdot \frac{R_{bulk_i}}{4} \tag{3}$$

$$R_y = \left[\beta \cdot (\sqrt{\frac{W}{L}} - 1) + 1\right] \cdot \frac{R_{bulk_i}}{4} \tag{4}$$

Here L is the x dimension and W is the y dimension. If $L = W$, the entire β term cancels and $R_x = R_y = R_{bulk_i}/4$. Empirically fitting to our FEM yields $\beta = 0.12$, which is independent of block sizes, aspect ratios, chip/package thickness, power density, and technology. After R_x and R_y are calculated, they need to be further subdivided if the block has more than one neighbor in each of the four directions.

This model is boundary- and initial-condition independent by construction since the component values are derived from material physical and geometrical values. Although there are several empirical factors used in the formulas (s, α, β), their values are the same, independent of the details of the chip and architecture being studied, within the limits of current state-of-the-art processes. Because the HotBlocks RC network is not obtained from full solutions of the time-dependent heat-conduction equation for every possible architecture, there may be boundary and initial condition sets that would lead to inaccurate results. But for the range of reasonable architectural floorplans and the short time scales of architecture simulations, the parameterized derivation of the chip model ensures that the model is indeed BICI for practical purposes if not in a formal sense. In other words, architects can rely on the model to be independent of boundary and initial conditions for purposes or architectural simulation.

A final note regards the temperature-dependence of conductivity. This temperature dependence only affects results for relatively large temperature ranges greater than 50° [27]. Our current chip model has a much smaller range in active mode, so we use the thermal conductivity of silicon at 85°, our specified maximum junction temperature. Incorporating a temperature dependence will be a necessary extension to HotSpot for working with larger temperature ranges.

8 Calibrating the Model

Dynamic compact models are well established in the thermal-engineering community, but we are unaware of any that have been developed to model temperature in a way that is compatible with typical microarchitecture research tools. Of course, the exact compact model must be validated to minimize errors that might arise from modeling heat transfer using lumped elements.

This is difficult, because we are not aware of any source of localized, time-dependent measurements of physical temperatures that could be used to validate our model. It

remains an open problem in the thermal-engineering community how to obtain such direct measurements. Eventually, we hope to use thermal test chips (e.g., [28]), and possibly also a system like an infrared camera or IBM's PICA picosecond imaging tool [29] with real superscalar processors to image heat dissipation on a fine temporal and spatial granularity. Until this becomes feasible, we are using a thermodynamics and computational fluid-dynamics tool as a semi-independent model which we can use for validation and calibration. As mentioned earlier, we are using Floworks [26], a commercial, finite-element simulator of 3D fluid and heat flow for arbitrary geometries, materials, and boundary conditions.

Floworks and HotSpot are only semi-independent, because some of the HotSpot parameters were calibrated independently using Floworks. Nevertheless, we can verify that the two obtain similar steady-state operating temperatures and transient response.

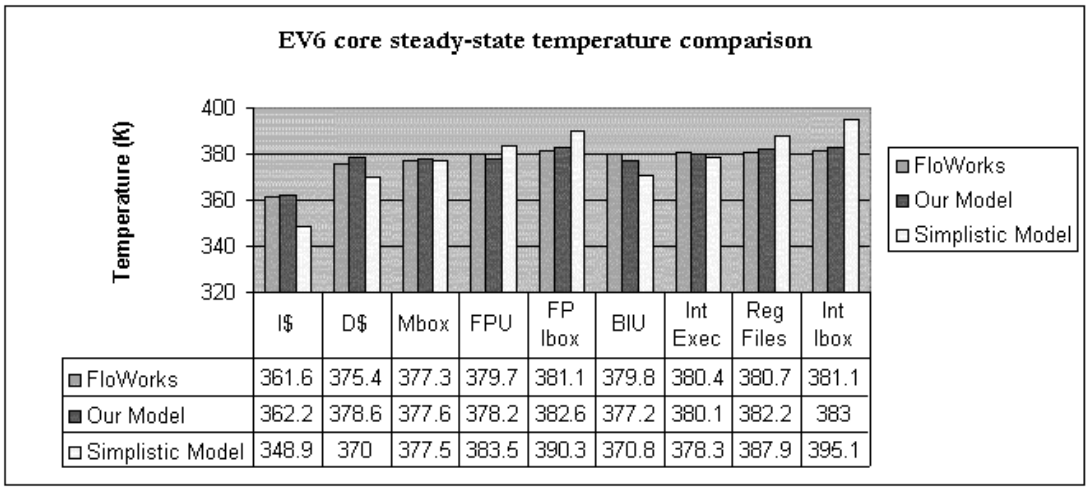
Figure 5a shows steady state validation comparing temperatures predicted by Floworks, HotSpot, and a "simplistic" model that eliminates the lateral portion of the RC circuit (but not the package) to show the importance of transfer impedances even for architecture studies. HotSpot shows good agreement with Floworks, with errors [1] always less than 5.6% and usually less than 2.5%. The simplistic model, on the other hand, has much larger errors, as high as 29%, plus has the drawback that the deviations are neither consistently high or low. One of the largest errors is for the hottest block, which means too many thermal triggers will be generated. Figure 5b shows transient validation comparing, for Floworks and HotSpot, the evolution of temperature in one block on the chip over time. The agreement is excellent between Floworks and HotSpot, but the simplistic model shows temperature rising much too fast. The agreement would be even worse had we omitted the package model.

Because the model is derived directly from the processor's floorplan and basic material properties of its package, it is easily added to any architectural simulator. This model is a portable module that is now publicly available on the web.

Due to the lack of an architectural temperature model, a few prior studies have attempted to model temperatures by averaging power dissipation over a window of time. This will not capture any localized heating unless it is done at the granularity of on-chip blocks, but even then it fails to account for lateral coupling among blocks, the role of the heat sink, etc. We have also encountered the fallacy that temperature corresponds to instantaneous power dissipation. This is clearly not true; the thermal capacitance acts as a low-pass filter in translating power variations into temperature variations.

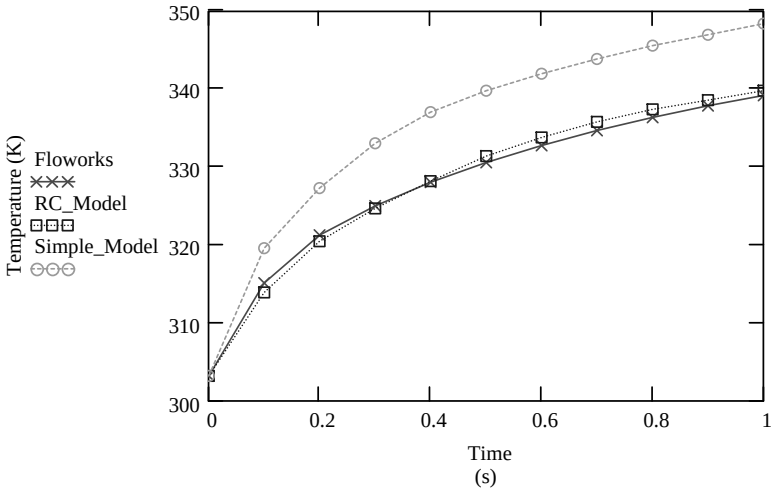
To show the importance of using a thermal model instead of a power metric, we have computed the correlation coefficient between temperature and the moving average of power dissipation. The data in Table 3 come from the *gcc* benchmark,

[1] Percent error in temperature is computed with respect to the ambient, $45°$C or $318°$K.



	I$	D$	Mbox	FPU	FP Ibox	BIU	Int Exec	Reg Files	Int Ibox
FloWorks	361.6	375.4	377.3	379.7	381.1	379.8	380.4	380.7	381.1
Our Model	362.2	378.6	377.6	378.2	382.6	377.2	380.1	382.2	383
Simplistic Model	348.9	370	377.5	383.5	390.3	370.8	378.3	387.9	395.1

(a) Steady state



(b) Transient

Fig. 5. Model validation. (a): Comparison of steady-state temperatures between Floworks, HotSpot, and a "simplistic" model with no lateral R and C. (b): Transient validation—this compares the step response in Floworks, HotSpot, and the simplistic model for a single block, the integer execution register file.

which is representative of all of the benchmarks which we examined. Temperatures were collected using HotSpot, and power measurements were collected using an averaging period of one million cycles, both for a simulation of one billion cycles. R-squared values for the correlation are reported in Table 3 along with the units' mean power and temperatures over the simulation, and the percent of time the units spend in thermal violation. The R-squared value gives the percentage of variance that is common between two sets of data: correlation is very poor for most units, especially the hot ones.

Table 3
Correlation of power and temperature for gcc.

Units	Temp (°C)	Power (W)	R-Square (%)	% cycles in thermal violation
I-Cache	78.3	4.53	16	0
BPred	78.2	1.94	26	0
ITB	79.1	0.21	24	0
IntExec	81.7	3.85	30	33
IntReg	83.2	2.97	20	60
IntQ	78.1	0.20	15	0
IntMap	76.3	0.58	08	0
LdStQ	79.7	1.84	16	0
D-Cache	83.1	8.44	15	57
DTB	79.1	0.11	06	0
FPMap	73.6	0.03	02	0
FPAdd	76.1	0.37	79	0
FPQ	77.0	0.01	78	0
FPMul	74.5	0.39	65	0
FPReg	75.4	0.23	51	0
L2	68.7	0.94	01	0

9 Conclusions and Future Work

This paper describes an approach to modeling thermal behavior in architecture-level power/performance simulators. It is based on a simple network of thermal resistances and capacitances, which are derived using area and floorplan information for the major functional blocks at the architecture level, and estimates of thermal resistance and capacitance for the microprocessor's thermal package. We describe techniques for automating the derivation of area, floorplan, package, and thermal R/C information to minimize the effort required of the processor architect. Area and floorplan information and the resulting thermal model must be derived using high-level estimation because architecture-level studies are often conducted before creating the lower-level processor descriptions that are needed by traditional algorithms. Package information can be derived using well-known methods for computing thermal resistance and capacitance for heat spreaders, heat sinks, convection, etc.

Early work using a simpler version of our model [4] showed the importance of modeling heat at finer granularities than the whole die, which is the approach that earlier work in the processor-architecture community had taken. This paper also shows the importance of modeling transfer or "lateral" thermal impedances. The thermal model which we propose here (HotSpot) therefore fills a void in the architecture community's modeling capabilities. It can be used to simulate how thermal stress is correlated to the architecture, how design decisions influence thermal behavior and other important effects that are dependent on operating temperature like leakage currents (for example, in conjuction with temperature-aware leakage models [30]), and for design of architectural runtime thermal management. HotSpot is available on the web at http://lava.cs.virginia.edu/hotspot.

Future work consists of developing models for sensor accuracy and delay (needed for realistic modeling of run-time thermal management) and more thorough validation of the model using physical measurements, either via test chips or thermal-imaging techniques. Another interesting avenue for future work is to find ways to adapt the more rigorous techniques of direct construction of dynamic compact models from numerical or analytical thermal models to describe the 3D heat flow with less approximation but in ways that are compatible with early-stage, high-level architecture studies.

It is our hope that this work will stimulate greater collaboration among the package, VLSI, and architecture communities to find better ways to manage heat at all levels of computer systems design.

Acknowledgments

We would like to thank Pradip Bose, David Brooks, George Cai, Howard Davidson, Ed Grochowski, and Margaret Martonosi for their helpful comments. We also thank Amar Dwarka, Yingmin Li, Yong Ma, Amit Naidu, Dharmesh Parikh, Paolo Re, Garrett Rose, Ram Suryanarayan, Hao Zhang, Yan Zhang and David Tarjan for their contributions to this work. This work is supported in part by the National Science Foundation under grants CCR-0133634, MIP-9703440, a grant from Intel MRL, and an Excellence Award from the Univ. of Virginia Fund for Excellence in Science and Technology.

References

[1] D. Brooks, M. Martonosi, Dynamic thermal management for high-performance microprocessors, in: Proceedings of the Seventh International Symposium on High-Performance Computer Architecture, 2001, pp. 171–82.

[2] W. Huang, J. Renau, S.-M. Yoo, J. Torellas, A framework for dynamic energy efficiency and temperature management, in: Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture, 2000, pp. 202–13.

[3] E. Rohou, M. Smith, Dynamically managing processor temperature and power, in: Proceedings of the 2nd Workshop on Feedback-Directed Optimization, 1999.

[4] K. Skadron, T. Abdelzaher, M. R. Stan, Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management, in: Proceedings of the Eighth International Symposium on High-Performance Computer Architecture, 2002, pp. 17–28.

[5] W. Batty, et al., Global coupled EM-electrical-thermal simulation and experimental validation for a spatial power combining MMIC array, IEEE Transactions on Microwave Theory and Techniques (2002) 2820–33.

[6] G. Digele, S. Lindenkreuz, E. Kaspar, Fully coupled dynamic electro-thermal simulation, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 5 (3) (1997) 250–57.

[7] V. Koval, I. W. Farmaga, MONSTR: A complete thermal simulator of electronic systems, in: Proceedings of the 31st Design Automation Conference, 1994.

[8] S. S. Lee, D. J. Allstot, Electrothermal simulation of integrated circuits, IEEE Journal of Solid-State Circuits 28 (12) (1993) 1283–93.

[9] M. N. Sabry, A. Bontemps, V. Aubert, R. Vahrmann, Realistic and efficient simulation of electro-thermal effects in VLSI circuits, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 5 (3) (1997) 283–89.

[10] V. Székely, A. Poppe, A. Páhi, A. Csendes, G. Hajas, Electro-thermal and logi-thermal simulation of VLSI designs, IEEE Transactions on VLSI Systems 5 (3) (1997) 258–69.

[11] K. Torki, F. Ciontu, IC thermal map from digital and thermal simulations, in: Proceedings of the 2002 International Workshop on THERMal Investigations of ICs and Systems (THERMINIC), 2002, pp. 303–08.

[12] S. Gunther, F. Binns, D. M. Carmean, J. C. Hall, Managing the impact of increasing microprocessor power consumption, in: Intel Technology Journal, 2001.

[13] D. Brooks, V. Tiwari, M. Martonosi, Wattch: A framework for architectural-level power analysis and optimizations, in: Proceedings of the 27th Annual International Symposium on Computer Architecture, 2000, pp. 83–94.

[14] M.-N. Sabry, Dynamic compact thermal models: An overview of current and potential advances, in: Proceedings of the 8th Int'l Workshop on THERMal INvestigations of ICs and Systems, 2002, invited paper.

[15] A. Krum, Thermal management, in: F. Kreith (Ed.), The CRC handbook of thermal engineering, CRC Press, Boca Raton, FL, 2000, pp. 2.1–2.92.

[16] P. Shivakumar, N. P. Jouppi, Cacti 3.0: An integrated cache timing, power and area model, Tech. rep., Compaq Western Research Laboratory (Feb. 2001).

[17] J. M. Mulder, N. T. Quach, M. J. Flynn, An area model for on-chip memories and its application, IEEE Journal of Solid-State Circuits 26 (2) (1991) 98–106.

[18] J. M. Rabaey, Digital Integrated Circuits: A Design Perspective, Prentice-Hall, Englewood Cliffs, NJ, 1995.

[19] M. Nemani, F. N. Najm, High-level area and power estimation for VLSI circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 18 (6) (1999) 697–713.

[20] D. F. Wong, D. L. Liu, A new algorithm for floorplan design, in: Proccedings of the Design Automation Conference, 1986, pp. 101–107.

[21] MIPS R10000 die photo, from website: CPU Info Center, http://bwrc.eecs.berkeley.edu/CIC/die_photos/#mips.

[22] Intel Pentium die photo, from website: CPU Info Center, http://bwrc.eecs.berkeley.edu/CIC/die_photos/pentium.gif.

[23] M. Matson, et al., Circuit implementation of a 600MHZ superscalar RISC microprocessor, Computer Design: VLSI in Computers and Processors 26 (2) (1998) 104–110.

[24] R. Mahajan, Thermal management of CPUs: A perspective on trends, needs and opportunities, keynote presentation at the 8th Int'l Workshop on THERMal INvestigations of ICs and Systems (Oct. 2002).

[25] Standard Performance Evaluation Corporation, SPEC CPU2000 Benchmarks, http://www.specbench.org/osg/cpu2000.

[26] Floworks: Fluid Flow Analysis for SolidWorks, NIKA GmbH. Website, http://www.floworks.com.

[27] M. Rencz, V. Székely, Studies on the error resulting from neglecting nonlinearity effects in dynamic compact model generation, in: Proceedings of the 8th Int'l Workshop on THERMal INvestigations of ICs and Systems, 2002, pp. 10–16.

[28] Z. Benedek, B. Courtois, G. Farkas, E. Kollár, S. Mir, A. Poppe, M. Rencz, V. Székely, K. Torki, A scalable multi-functional thermal test chip family: Design and evaluation, Transactions of the ASME, Journal of Electronic Packaging 123 (4) (2001) 323–30.

[29] M. McManus, S. Kasapi, PICA watches chips work, Optoelectronics World .

[30] D. Parikh, Y. Zhang, K. Sankaranarayanan, K. Skadron, M. Stan, Comparison of state-preserving vs. non-state preserving leakage control in caches, in: Proceedings of the 2003 Workshop on Duplicating, Deconstructing, and Debunking, 2003.

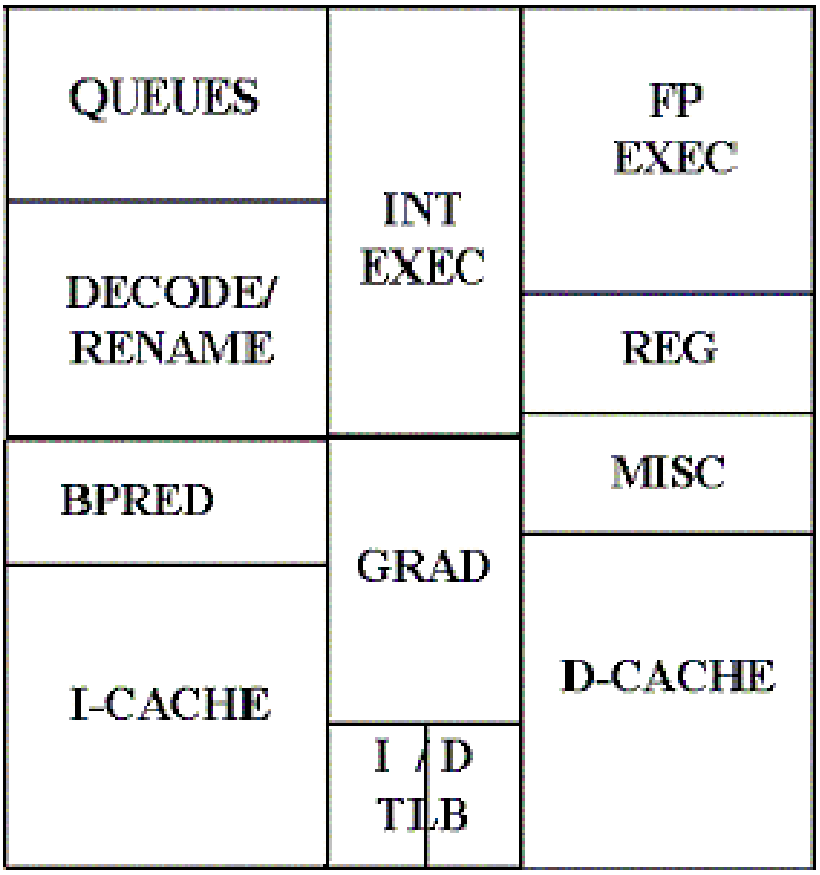


Fig. 1. The semicircular floorplan generated by the HotFloorplan tool using the MIPS R10000 configuration.

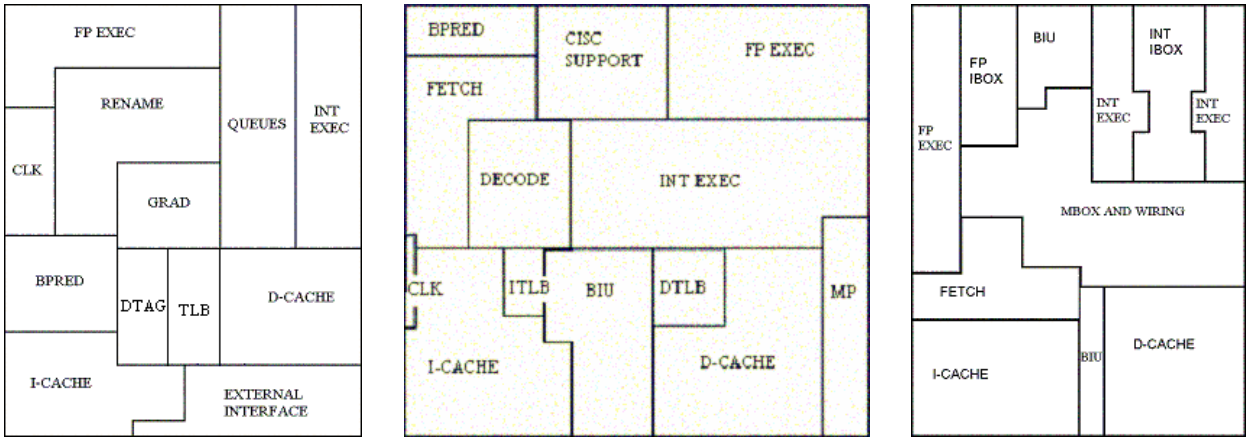
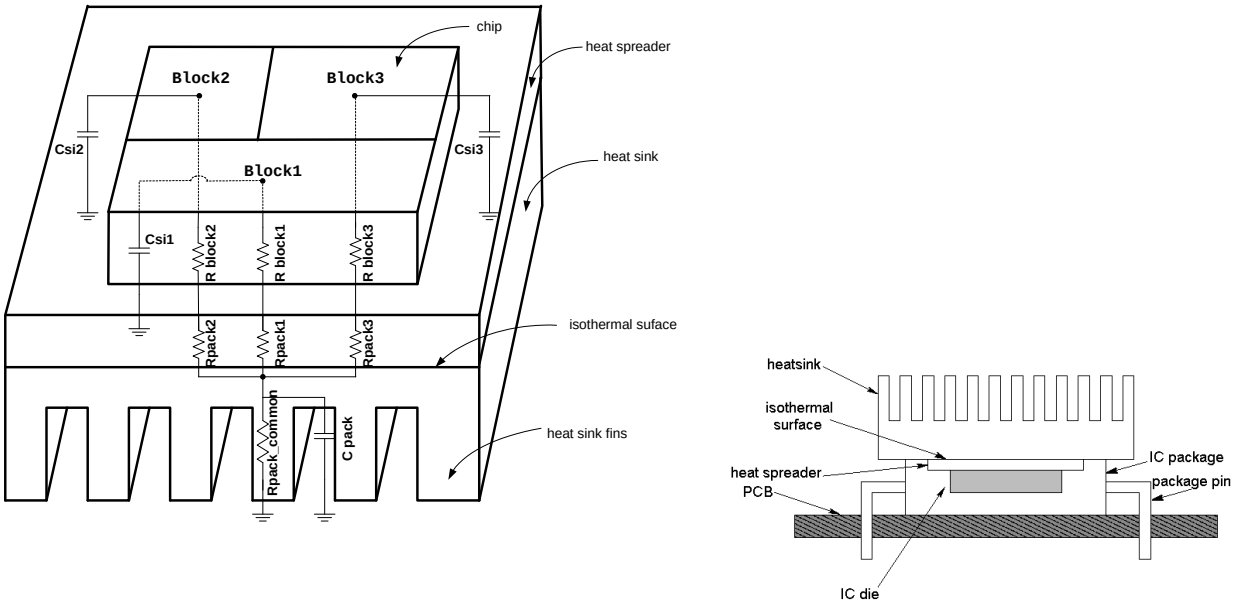
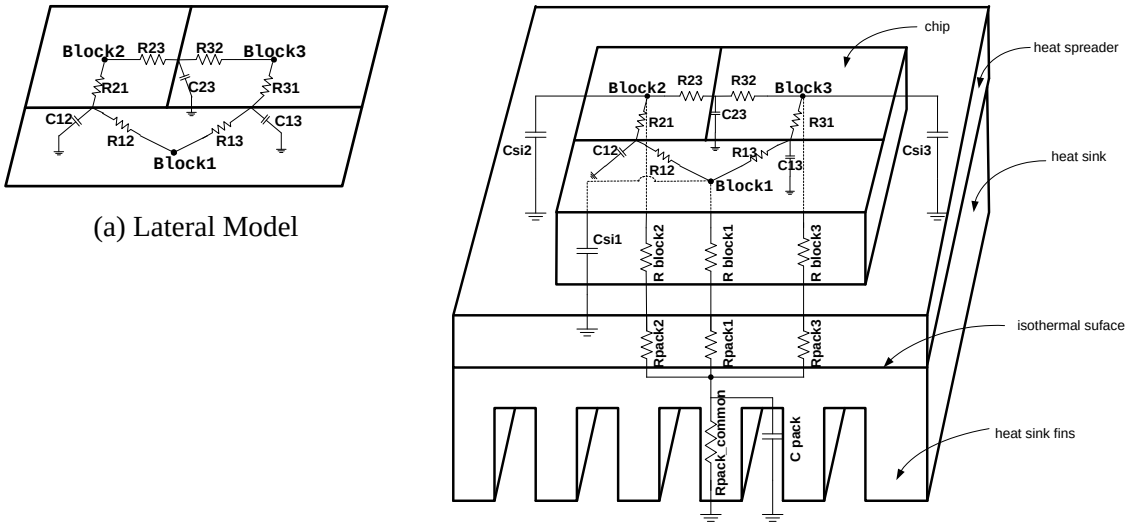


Fig. 2. “Normalized” floorplan for the MIPS R10000 (left), Intel Pentium (middle), and Alpha 21264 (right) processors.



(a) Vertical model (b) Typical Package

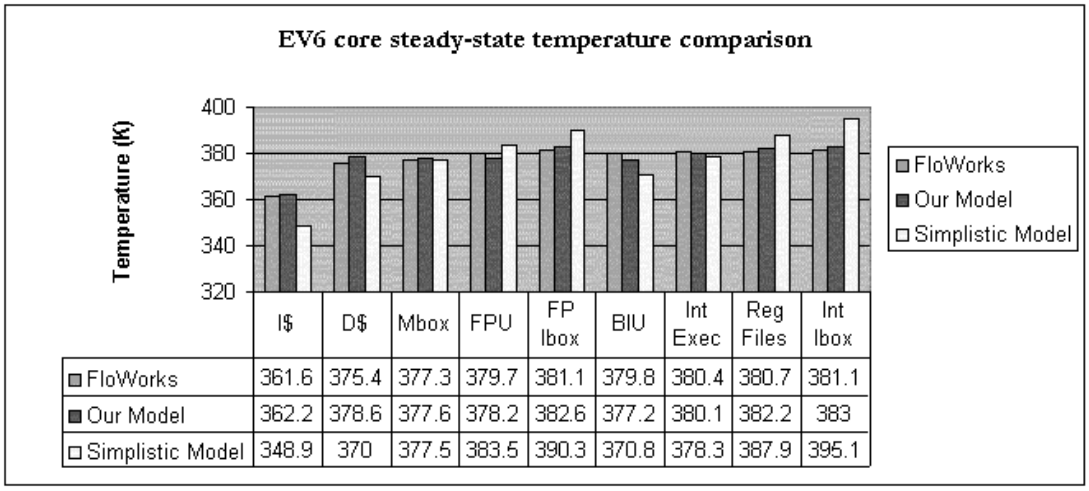
Fig. 3. (a): Example vertical RC network, with an RC pair from the center of each unit to the thermal package; an RC pair for each corresponding block of the package; an RC pair for each corresponding block of the heat sink above the isotherm; and a shared RC pair for the thermal package below the isotherm. For clarity, this image is upside-down relative to the typical use as in (b): Side view of a typical package configuration upon which the HotSpot model is based. The model combines the heat spreader and heat sink into a single unit.



(a) Lateral Model

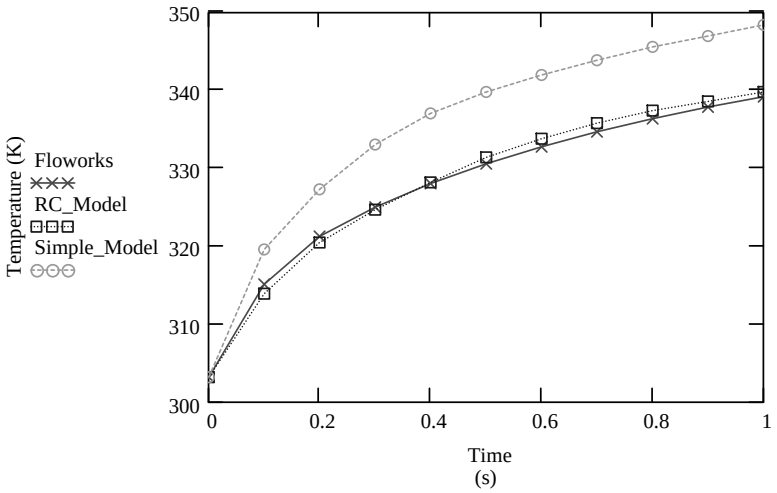
(b) Complete Model

Fig. 4. (a): Lateral model, with an RC pair between the center of each node and the center of each shared edge. (b):Overall RC model, including both vertical and lateral components.



	I$	D$	Mbox	FPU	FP Ibox	BIU	Int Exec	Reg Files	Int Ibox
FloWorks	361.6	375.4	377.3	379.7	381.1	379.8	380.4	380.7	381.1
Our Model	362.2	378.6	377.6	378.2	382.6	377.2	380.1	382.2	383
Simplistic Model	348.9	370	377.5	383.5	390.3	370.8	378.3	387.9	395.1

(a) Steady state



(b) Transient

Fig. 5. Model validation. (a): Comparison of steady-state temperatures between Floworks, HotSpot, and a “simplistic” model with no lateral R and C. (b): Transient validation—this compares the step response in Floworks, HotSpot, and the simplistic model for a single block, the integer execution register file.

Table 1
Distances from computed vs. observed locations of the major functional blocks. Results are in percent, normalized to the height of the chip.

	Alpha	MIPS	Pentium
I-cache	6.4%	5.4%	11.1%
D-cache	3.9%	10.8%	9.1%
Integer execution	NA	36.8%	24.8%
FP execution	71.2%	49.1%	5.6%
Queues	NA	50.8%	21.7%
Branch pred.	4.4%	7.9%	52.3%
Decode/rename	37.9%	22.7%	7.1%

Table 2
Published and calculated values for thermal resistance for four heat sinks. Published values are taken from http://www.allstarcomponents.com/cooling.html. “Rect” means rectangular.

Material	Area (cm^2)	Base Thick (cm)	# Fins	Fin Type	Fin Ht. (cm)	Air Sp. (m/s)	Spec. R (K/W)	Calc. R (K/W)
Al	36	0.69	28	Rect	2.06	5	0.67	0.613
Cu	61	1.17	256	Pin	1.68	10	0.36	0.38
Al	64	0.60	18	Rect	2.4	8	0.60	0.62

Table 3
Correlation of power and temperature for gcc.

Units	Temp (°C)	Power (W)	R-Square (%)	% cycles in thermal violation
I-Cache	78.3	4.53	16	0
BPred	78.2	1.94	26	0
ITB	79.1	0.21	24	0
IntExec	81.7	3.85	30	33
IntReg	83.2	2.97	20	60
IntQ	78.1	0.20	15	0
IntMap	76.3	0.58	08	0
LdStQ	79.7	1.84	16	0
D-Cache	83.1	8.44	15	57
DTB	79.1	0.11	06	0
FPMap	73.6	0.03	02	0
FPAdd	76.1	0.37	79	0
FPQ	77.0	0.01	78	0
FPMul	74.5	0.39	65	0
FPReg	75.4	0.23	51	0
L2	68.7	0.94	01	0