The Thesis Committee for Darshan Dhimantkumar Gandhi
Certifies that this is the approved version of the following thesis:

Core Level Thermal Estimation Techniques for Early Design Space Exploration

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Lizy Kurian John

Co-Supervisor:

Andreas Gerstlauer

Core Level Thermal Estimation Techniques for Early Design Space Exploration

by

Darshan Dhimantkumar Gandhi, B.E.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

The University of Texas at Austin

May, 2014

Dedication

This thesis is dedicated to my beloved parents and to my friend, Saurabh whose enthusiasm and support led me to research activities.

Acknowledgements

I would like to thank Dr. Lizy John for her help and guidance on this thesis. It was through her teachings that I was introduced to the concepts of computer architecture and performance-power evaluation techniques. I would like to thank my co-supervisor Dr. Andreas Gerstlauer for his rigorous support on the research problem and constructive feedback. Finally, I would like to thank my research group members for providing useful suggestions during my research.

Abstract

Core Level Thermal Estimation Techniques for Early Design Space Exploration

Darshan Dhimantkumar Gandhi, M.S.E.

The University of Texas at Austin, 2014

Supervisor: Lizy Kurian John

Co-supervisor: Andreas Gerstlauer

The primary objective of this thesis is to develop a methodology for fast, yet accurate temperature estimation during design space exploration. Power and temperature of modern day systems have become important metrics in addition to performance. Static and dynamic power dissipation leads to an increase in temperature, which creates cooling and packaging issues. Furthermore, the transient thermal profile determines temperature gradients, hotspots and thermal cycles. Traditional solutions rely on cycle-accurate simulations of detailed micro-architectural structures and are slow. The thesis shows that the periodic power estimation is the key bottleneck in such approaches. It also demonstrates an approach (FastSpot) that integrates accurate thermal estimation into existing host-compiled simulations. The developed methodology can incorporate different sampling-based thermal models. It achieves a 32000x increase in simulation throughput for temperature trace generation, while incurring low measurement errors (0.06 K- transient, 0.014 K- steady-state) compared to a cycle-accurate reference method.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Power and temperature of modern day systems have become important metrics in addition to performance. Recent advancements in electronics, such as handheld devices and wearable electronic devices have promoted power efficient designs over performance oriented designs. This paradigm shift is mainly attributed to technology scaling, limited power supplies on small, mobile devices and lack of efficient cooling solutions. These devices primarily operate using power from lithium-ion batteries, which is a limited and expensive power source. Therefore, it is required to use the power supply sparingly, leading to minimal power consumption while providing maximum possible performance. Krisztian Flautner, a researcher from ARM Holdings remarks that, "Everyone wants a supercomputer in their pocket that is always on, communicating and never requires a charge."

The channel length of semiconductor devices is shrinking rapidly, to keep up with Moore's law [18] [34]. Beyond 11 nm length, quantum phenomena in the transistor can affect transistor operation substantially. At small process technologies, the effects of electro migration, hot carrier injection, negative-bias temperature instability (NBTI) and leakage current become more pronounced. These effects are influenced by the device temperature. In order to perform an accurate quantitative study of such effects, thermal characterization of the hardware becomes very important.

Cooling and packaging solutions directly impact heat dissipation capability of the integrated circuits (ICs) [24] [25]. But, the transistor density on semiconductor devices is also increasing rapidly. Designing the cooling mechanism for the worst case temperature scenarios has become extremely costly. Moreover, the popular solutions, such as heat

sinks and cooling fans are either not applicable to mobile devices or can only be used to the minimal effect due to form factor limitations. In addition, wearable electronic devices often place stricter constraints on the thermal responses of these systems, because of the surrounding surfaces include human skin or equivalent live medium. A system without a proper mechanism for power and thermal management is deemed to be poor. These effects require chip architects to design a system, which is power and thermal aware and can change its behavior accordingly. Therefore, chip architects explore run time hardware and software-based dynamic thermal management (DTM) techniques [26] [27]. An accurate analysis of the thermal response of the processors can be used to guide upcoming cooling/packaging approaches and study effects of DTM policies.

In the early stage of design exploration, detailed information about power and thermal characteristics of the system is not available. As outlined above, the need for accurate and early thermal estimation motivates the development of power and thermal simulation frameworks. Moreover, such simulations can provide quick tradeoff analysis during early design phase. A typical solution in architecture domain is to use a representative model of the chip, a power estimator tool and a thermal estimation tool in order to characterize the temperature. The representative model (usually an instruction set simulator (ISS)) can provide access/occupancy statistics, quantifying the activity factor of various components. This information is used by an approximated power estimation tool (examples, [1] [9] [23]) to calculate dynamic and leakage power. A periodic repetition of power estimation for different time intervals creates a specific form of power trace, which essentially denotes power consumption in modeled chip components over time. A thermal estimator tool (examples, [14] [16] [17]) is finally deployed to find out transient and steady-state temperature profiles of the chip for a given application.

Development of an effective thermal management scheme is inherently an iterative and time consuming task. Multiple benchmarks are run using detailed simulation to understand behavior across a range of optimizations. Along with rapid advancements in computer architecture, the benchmarks to compare upcoming processors are becoming complex and large. For example, the SPEC CPU 2006 [20] benchmark suite consists of 29 benchmarks with dynamic instruction counts in the hundreds of billions [28]. Similarly, a mobile computing benchmark suite, such as MiBench [6] [11] has millions of dynamic instructions per benchmark. Simulating these applications for accurate thermal characterization may take multiple days and huge compute resources. Hence, there is a need for a quick, accurate and integrated approach for thermal analysis.

The objective of this thesis is to develop a novel thermal estimation methodology called FastSpot, to improve the temperature estimation approach. Many architectural level temperature estimation tools [14] [16] [17] that accept an existing power trace have been developed in the research community, but there has been limited support for an integrated methodology. This thesis shows that the primary bottleneck in an integrated setup is periodic estimation of power to be fed into temperature estimation tools. The FastSpot methodology leverages host-compiled back-annotation approaches [2] [29] [30] to create approximate and online models for computing power traces. When these computation models are integrated with available thermal models, power estimation overhead during benchmark execution is reduced, resulting in significant speedup in simulation.

This thesis is organized as follows. Chapter 2 describes previous work that has been done in three major relevant areas, namely thermal estimation models, integrated simulation methodologies and host-compiled approaches. Chapter 3 describes the FastSpot methodology in details. The back-annotation setup, thermal models and their

integration is discussed in the chapter. The reference methodology, developed to evaluate FastSpot is described in Chapter 4. Chapter 5 elaborates on the evaluation setup for experiments and shows speed and accuracy tradeoffs of the proposed methodology. One important application of FastSpot is characterization of hotspots during design space exploration. Comparisons regarding hotspot estimation are provided in Chapter 6. Chapter 7 lists several directions on extending the work and Chapter 8 concludes the findings of this thesis.

Chapter 2: Related Work

Various research groups have provided important contributions to mitigate thermal simulation complexities during past few years. The contributions are classified in three major categories, as described below.

2.1 THERMAL AND POWER ESTIMATION MODELS

One important aspect of a thermal measurement methodology is the temperature estimation tool. These tools typically accept a certain type of power trace and calculate temperature based on various mathematical approaches. The tools also require the user to provide details of the material, including its thermal characteristics, ambient conditions and a spatial distribution of various power sources.

One of the pioneering works in this direction is HotSpot [14]. It dynamically builds a representative RC model of the chip based on a given floorplan, and it computes the temperature profile of the chip over a sequence of time stamps (called sampling periods) based on a given trace of power dissipation values. It uses numerical analysis methods (fourth order Runge-Kutta) for calculating temperature. HotSpot requires a structurally accurate power trace over all components in the floorplan at the chosen sampling period.

Discrete time temperature evaluation models (DTTEMs) [8] [15] are an approximated set of equations, which model first order effects of temperature. The equations are linear, making them computationally inexpensive. Temperature is modeled as a function of current power and current temperature. For sufficiently small sampling periods, such methods are expected to yield accurate results. A brief mathematical analysis of such a model is presented in Chapter 3.

SESCTherm (SuperESCalar simulator thermal model) [31] is a temperature modeling framework, aiming to include thermal effects of various components of a typical semiconductor device. The model establishes the importance of components such as, mainboard, package, interconnects and recent manufacturing techniques, such as silicon on insulator (SOI). The SESCTherm modeling framework integrates thermal models for interconnect, mainboard, package and silicon-on-insulator transistors. In addition to these models, a simple model to account for temperature based variations in material properties is also integrated. The claim of the research is that if transient temperatures are being measured for a long duration, a temperature model built without considering such effects is likely to produce far from actual behavior. The example illustrated in the paper demonstrates a case where simple thermal models will produce an error of 30-40 C with respect to actual hardware behavior of the chip due to lack of advanced technology related thermal models.

The so-called Power blurring (PB) [32] technique can model transient and steady-state temperature profiles for chips. The PB method attempts to base thermal analysis on image blurring methods, used for processing photographic images. In this method, power distribution is considered as the raw image and temperature response is assumed to be a blurred version of the power map. A commercial tool, ANSYS [33], is used to perform finite element analysis (FEA) on the power map to find its impulse response. HotSpot, SESCTherm and Power blurring methods are compared for steady-state and transient temperature trace generation [17]. The authors conclude that a PB method provides accurate temperature profiles with lower execution time requirement.

Integrated space and time adaptive chip-package (ISAC) [16] is a dynamically adaptive thermal analysis technique. Spatial adaptation is facilitated by hybrid oct-tree

data structures, which can be used for efficient management of spatial nodes and their processing. A time marching algorithm is employed to achieve adaptation in time, which involves iteratively advancing local times while solving a set of partial differential equations simultaneously. ISAC aims to change time and space granularity to make thermal computations faster while minimizing the impact on accuracy. The Fast asynchronous time marching technique (FATA) extends the time marching algorithm to allow different nodes having their own local time steps for improve speed.

Time invariant thermal linear system (TILTS) [7] improves temperature estimation speed by a factor of 1300 over HotSpot. It explores opportunities in the power trace for performing linear computations, rather than solving for temperature using numerical methods at every estimation interval. Redundant integral calculations are replaced with linear computations, which results in a considerable speed up. An improved algorithm Convolutional TILTS (CONTILTS) is also proposed to overcome unnecessary floating point operations, resulting in a 6000x speedup over HotSpot. All the mentioned models work on an arbitrary form of a power trace, which is a record of power consumption in floorplan components over time. Generation of such power traces and the associated complexity is not considered in the abovementioned methods.

In order to generate a power trace, power consumption of the chip is repeatedly calculated using a power estimator. Many industrial and research tools are available to address power computation at lower levels, i.e. after floor planning and layout is completed. Some of the examples are PowerMill™ [21] and QuickPower™ [35]. These tools cannot be efficiently used at making decisions during architectural exploration. The need for a framework to analyze power at the architectural level has been addressed by various research groups. Wattch [1] is one of the pioneering works for integrating a cycle accu-

rate simulator with power estimation models. It uses the Simplescalar [22] simulator as a front end and parses the hardware access counts during an application execution. These access counts are used by power models of various types of architectural blocks such as, clocking, arrays, associative structures and wiring. The power numbers are reported to be within 10% error of the lower level models.

Orion [23] is a power model for networks on chip (NoCs). It provides models for calculating area, dynamic power and leakage power in NoC-based environments. Overall chip floorplan details and short circuit power are not modeled in version 2.0. The CACTI [36] tool is developed to estimate delays in various configurations. CACTI can model uniform and non-uniform cache access and assess hit/miss latencies and other cache metrics by breaking down cache hierarchies into small and regular structures, e.g. arrays, wires and content addressable memories (CAMs). Multicore power, area and timing (McPAT) [9] tool is based on CACTI models. McPAT accepts a configurable XML file specifying process, architectural and timing related configurations from the user. Some of the typical input arguments are provided in Table 3.1 in Chapter 3. An architectural model is developed by determining sizes of constituent components of an architectural block. Total power consumption is calculated by the following equation [9]:

$$\mathbf{P_{total}} = P_{dynamic} + P_{short\text{-}circuit} + P_{leakage}$$

$$= \alpha C * V_{dd}\Delta V * F_{clk} + V_{dd} * I_{short\text{-}circuit} + V_{dd} * I_{leakage}$$

where α denotes the activity factor of a particular component, which is usually measured by detailed architectural simulations.

Dynamic power consumption is a function of activity factor, output capacitance, voltage and frequency of operation. McPAT uses ITRS projections to correlate technology process to operating voltage [34]. Therefore, voltage does not appear as a configurable

option in the XML file. The activity factor is calculated as the ratio of accesses to a particular component in a defined cycle window. For example, 10 accesses to the ALU block in 100 cycles achieve an activity factor of 10/100 or 0.1. Out of three power components, the thesis targets dynamic power consumption for temperature calculations.

2.2 SIMULATION METHODOLOGIES

Researchers have looked into developing an integrated methodology for temperature estimations. The methodologies aim to facilitate fast architectural exploration for classifying temperature effects.

Integrated thermal estimation procedures are useful in exploring scheduling and layout techniques in multi core processors [3]. The authors of [3] use Multi2Sim, a multi-core architectural simulator, in conjunction with McPAT and HotSpot tools to set up a methodology for hotspot determination. As the focus of their research is in design space exploration, bottlenecks in this integrated methodology are not quantified. This thesis adopts a reference (cycle-by-cycle) methodology from this work for bottleneck study and comparisons.

System level thermal emulator (SLTE) [15] is an approach that looks for better task allocation, binding and scheduling problems from a thermal perspective. The average power consumption and average case execution time is characterized for a piece of software on any given core. The power traces are generated from a segment-based power model, which can annotate power values for a task level timetable. SLTE approximates temperature evaluation to the first order, in order to improve system responsiveness. This methodology is deployed at higher abstraction levels. The details of average power calculation are not described in the work.

The need to power and thermal aware design space exploration has been addressed in [19]. Again, the objective of the research work is to look at various design tradeoffs involving L2 cache features, number of cores, issue width, etc. The experimental setup follows a similar infrastructure, where a CMP microarchitecture simulator (SESC), a set of power models (Wattch, Orion, CACTI) and thermal models (HotSpot) are bound together to generate thermal traces. The method of obtaining transient temperature trace for a set of benchmarks is fundamentally the same, where a power estimator periodically estimates dynamic power and feeds it to HotSpot.

2.3 HOST-COMPILED APPROACHES

FastSpot leverages concepts of host-compiled simulations in order to provide a fast and accurate methodology. Host-compiled models have emerged as an alternative to ISS-based approaches. In ISS-based models, micro architectural or instruction level models of computation are developed to describe the processor or the system. Such models are slow because of detailed computation structures. When a system-level model is being developed, there is a possibility to increase the level of abstraction from instructions to basic blocks or functions. Additionally, the communication also can be viewed at the level of words or messages, rather than at cycle granularity. A higher abstraction level results in faster simulation speed. Host-compiled models describe functional models in the form of source code itself, such that fast simulation speed is obtained by native compilation and execution. System-interactions are modeled through transaction level modeling (TLM), which can reduce unnecessary overhead for simulation of communication during early estimations. Performance and power metrics are obtained through simulations and fed into the host-compiled models. Features of host-compiled models and challenges are described in [5].

Abstract system level models using the host-compiled approach are developed in [37]. Application level back-annotation of timing is performed using micro architectural description language (uADL) models, and McPAT is used for power calculations. These models are integrated with abstract models of processors and real time operating system (RTOS), where communication is defined by TLM backplanes. With such integrated approaches, time spent in complete design space exploration (DSE) was shown to be six times lower.

The retargetable back-annotator (RBA) [2] tool demonstrates the speed and accuracy tradeoffs in timing and energy estimation with respect to traditional ISS-based approaches. The details of the back-annotation process, including basic block characterization for timing and energy are provided in [2]. RBA achieves very high simulation speeds (~2000 MIPS), which are on the order of pure functional simulations. The errors in measurement of energy and timing are less than 1%, when compared to ISS-based approaches. The errors are introduced because a basic block characterization phase cannot accurately identify all dynamic scenarios during actual application execution. Actual execution latency of a basic block is not only dependent on its predecessor, but on predecessors of predecessors, register and memory values, etc. The lack of such knowledge during characterization adds to measurement errors.

The proposed FastSpot methodology extends the RBA tool for transient and steady-state temperature measurements. It leverages back-annotation and characterization concepts, as they can expedite transient temperature generation at low loss in accuracy. The thesis explains steps in the back-annotation process in section 3.1.

Chapter 3: FastSpot Methodology

In this section, the overall flow of the proposed FastSpot methodology is demonstrated. Some of the prominent features of host-compiled simulation [2] that this work extends on are also explained. Figure 3.1 shows the block diagram of the proposed approach.

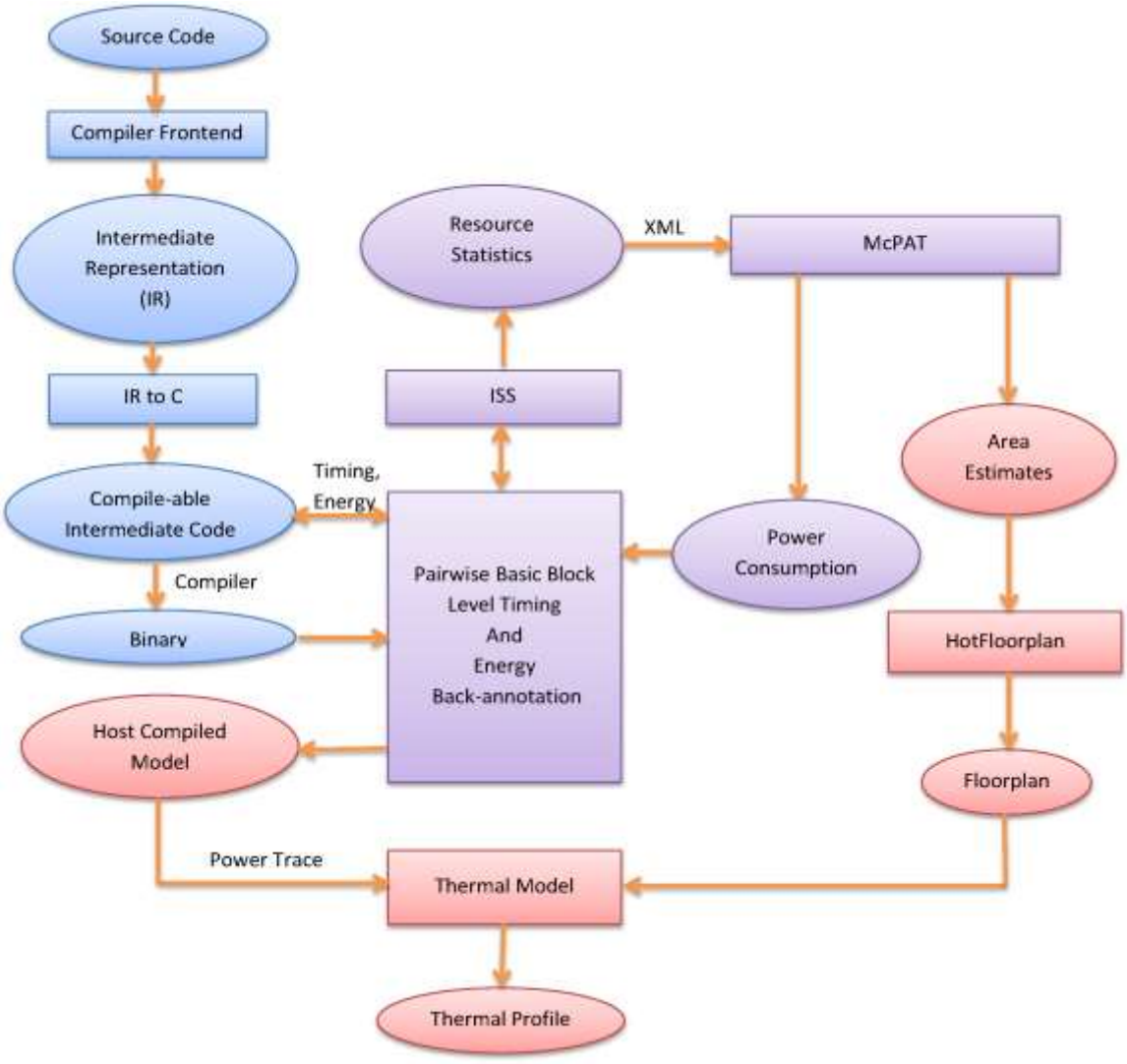

Figure 3.1: FastSpot Methodology Diagram.

The underlying concept of FastSpot is back-annotation at the intermediate representation (IR) level. Application source code is passed through a compiler to translate code into an IR that accurately reflects front-end optimizations. Following this, basic blocks (BBs) in the IR are annotated with their execution latency and energy, obtained using a retargetable, cycle-accurate ISS and McPAT models, respectively.

During trace analysis of ISS output, a set of access statistics is collected and subsequently fed into McPAT. McPAT provides dynamic power dissipation of each component using the activity information. Finally, the power dissipated during a block execution is converted into energy and annotated in the IR code. This is called the characterization phase, which includes a pair-wise characterization of each basic block with all its possible predecessors in order to accurately account for path- and history-dependent effects on latency and energy.

Temperature estimation tools, such as HotSpot and DTTEM are incorporated in the flow. Typically, thermal estimation tools require a form of power trace, which indicates power consumption in various components of the floorplan of a chip over time. FastSpot extends existing back-annotation approaches [2] [29] such that the annotated energy dissipation values in the intermediate representation of the source code can be used by the host-compiled model to generate a structurally accurate power trace. In order to model the spatial distribution of temperatures, estimation tools also require the area and placement of components of a chip. Such area and placement estimates of various blocks are obtained using McPAT and the hotfloorplan [14] utility. Combined, the power trace and the floorplan can be utilized by the integrated thermal model to generate the thermal profile over time.

A back-annotated intermediate representation of the source code, when integrated with the functional implementation of a thermal model results in FastSpot, a host-compiled model for temperature estimation. FastSpot achieves speedups due to faster power trace generation and application of light-weight temperature models, such as DTTEM. In the following, three aspects of FastSpot approach are discussed in more details: (1) organization of the back-annotation process, (2) employed thermal models and (3) integration of thermal models, making FastSpot a comprehensive temperature simulation methodology.

3.1 BACK-ANNOTATION PROCESS

The back-annotator tool works on the interfaces as shown in the figure 3.2.

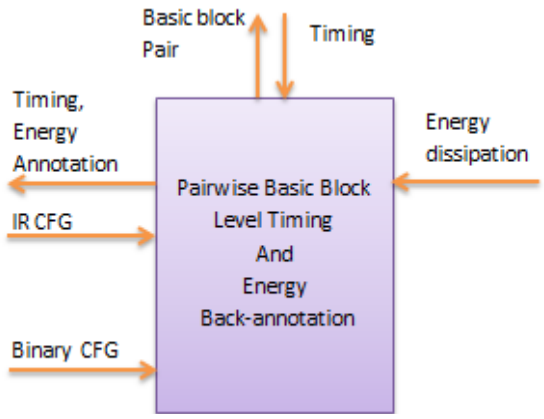

Figure 3.2: Back-annotator Tool Interface

Back-annotation process involves identification of granularity for annotation, which in this case is a basic block. The application is passed through a compiler and instructions that can change the control flow (branch, calls etc.) are identified. The entire control flow graph (CFG) is then segmented in basic block sets, which consist of prede-

cessor and successor pairs. The basic block characterization process involves running each unique basic block pair in the application CFG on reference models which can provide detailed trace of basic block execution and related metrics, such as timing and energy. It is important to note that the predecessor basic block is used to set up the internal pipelines of the ISS, so that accurate timing and resource information can be obtained. Access statistics and timing numbers do not involve the usage by predecessor blocks. The back-annotator tool maintains structures to store varying path dependent effects.

Execution latency (timing) of the basic block is obtained by simulating it on a cycle-accurate ISS, generated from uADL models from Freescale [4]. The instructions constituting a basic block are obtained from the object file of the cross-compiled application. The ISS accepts machine code of instructions and then executes them functionally, generating a detailed trace of internal pipeline stages of the architecture and the resultant register and memory values. Moreover, the input of the ISS can also include initial values of register and memory. These inputs are gathered from the output trace of the preceding basic blocks. The overall process is explained in figure 3.3.

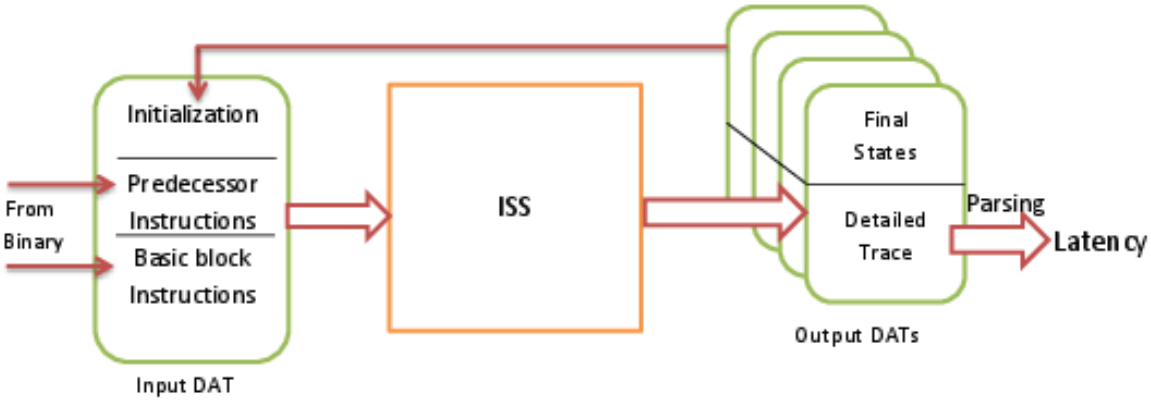

Figure 3.3: Latency Analysis Procedure

Energy consumption in various floorplan components is obtained using McPAT. McPAT uses an XML interface to feed required structural, operational and resource utilization statistics. The major knobs used in FastSpot setup are listed in Table 3.1. The output file is parsed to calculate dynamic power consumption and calculate energy dissipation using latency and operating frequency details.

Category	Parameters
Structural features	Machine width, Fetch/Issue/Decode width, Pipeline depth, Number of functional units, Registers, ROB size, Memory ports
Operational features	Technology node, Operating frequency, Ambient temperature
Resource utilization	Total cycles, Total instructions, Register accesses, Memory and integer instructions

Table 3.1: McPAT Input Parameters

```
//Latency and energy_values are obtained from
characterization phase
//The IR remains the same irrespective of thermal model
BB_i:
        //Block code
        j = j +1;
        end = (long unsigned int) p;
        if (j < 4) {
        goto BB_o;
        }
        else {
        goto BB_k;
        }
BB_k:
.....
```

Figure 3.4: Sample Intermediate Representation

The characteristic information about basic block pair is back annotated in the intermediate representation (IR) of the application. Figure 3.4 describes a sample structure of an application IR. The back-annotation process uses a script based framework to identify predecessors of the basic block. If there is a single predecessor, characterized information is appended as shown in figure 3.5. If there are multiple predecessors, script is responsible to generate an intermediate code that caters to all possible entry points to the basic block in CFG. A basic block with two predecessors will have a similar structure as figure 3.6 after being back-annotated. In figure 3.6, *BB_x* and *BB_y* represent the predecessors to the basic block *BB_i*. In each basic block, two string identifiers get updated namely, current basic block identifier (*curr_BB*) and predecessor identifier (*pred_BB*). *Pred_BB* is used to condition the shown if statement and is updated with *curr_BB* value after the 'if-statement' execution. Overall, this method is used to clearly identify predecessors of a basic block so that relevant information is used during run time.

```
.....
//Latency and energy_values are obtained from characterization
phase
//The IR remains the same irrespective of thermal model
BB_i:
        curr_BB = "BB_i";
        //Appended functions
        increment_latency (latency_i);
        increment_energy (energy_i);
        pred_BB =curr_BB;
        //Block code
        j = j +1;
        end = (long unsigned int) p;
        if (j < 4) {
                goto BB_o;
        }
        else {
                goto BB_k;
        }
BB_k:
.....
```

Figure 3.5: Annotated IR (Single Predecessor)

```
----------
//Latency and energy_values are obtained from characterization
phase
//The IR remains the same irrespective of thermal model
BB_i:
        curr_BB = "BB_i";
        //Appended functions
        if (pred_BB == "BB_x"){
        increment_latency (latency_x_i);
        increment_energy (energy_x_i);
        }
        else if  (pred_BB == "BB_y"){
        increment_latency (latency_y_i);
        increment_energy (energy_y_i);
        }
        pred_BB =curr_BB;
        //Block code
        j = j +1;
        end = (long unsigned int) p;
        if (j < 4) {
                goto BB_o;
        }
        else {
                goto BB_k;
        }
BB_k:
.....
```

Figure 3.6: Annotated IR (Multiple Predecessors)

One important problem in back-annotation is discrepancy between IR and the binary CFG. The back-annotator tool analyzes binary CFG and attempts to correlate characterized information in the IR. Due to compiler backend optimizations, IR and the binary may have different control flow graphs and some basic blocks may not have one to one mapping. One simple way to tackle the problem is to turn off back end optimizations, which will lead to poor performance. Another solution is to create a mapping table to

identify correlated basic blocks in the binary and the IR by inspection, which is slow and tedious process. FastSpot uses cost based optimization approach, as described in [2] to automatically map basic blocks in binary to basic blocks in the IR.

3.2 MONITORING FUNCTIONS

The annotated metrics are used by monitoring functions (*increment_latency()* and *increment_energy()*), as shown in Figure 3.7. These functions track and process elapsed cycles and dissipated energy for the application, as it progresses in time. The function *increment_energy()* simply increments a global array *current_energy[]* by the component-specific *energy_values[]* (obtained through characterization) of the current basic block. The function *increment_latency()* does similar updates for cycle counts. Moreover, the latter function is responsible for temperature estimation. Whenever elapsed cycles reach a multiple of the predefined sampling period, the function calculates energy dissipated in the previous sampling period by taking a difference of *current_energy* and *previous_energy* arrays. It also converts energy difference to power dissipation using knowledge of the sampling period length and operating frequency (function *calculate_power_array()*). The power values are used by *calculate_temperature()* in order to generate a temperature trace. This function is specific to the thermal model integrated into the flow. Section 3.3 describes two implementations of this function for the considered thermal models.

The back annotated application, along with the models is compiled on the host machine. When executed, it provides accurate functional results as well as interfaces characterization details, such as latency and energy dissipation information for target architecture with the thermal model to generate temperature traces.

```
//Global variables being updated
unsigned long cycle_count;
//NUM_COMPONENTS is defined to be
//total number of components in a floorplan
double current_energy[NUM_COMPONENTS];
double previous_energy[NUM_COMPONENTS];
double Power_values[NUM_COMPONENTS];
unsigned long counter;

//The basic operations of functions remain the same
//irrespective of the thermal model
----------
void increment_latency(unsigned int n) {
        cycle_count += n;
        If (cycle_count > sampling_period*counter) {
                counter++;
                Power_values = calculate_power_array();
                //The following method depends on the thermal model
                calculate_temperature(Power_values);
                for (i =0; i <NUM_COMPONENTS; i++) {
                        previous_energy[i] = current_energy[i];
                }
        }
}
void increment_energy(double m[]) {
        for (i =0; i <NUM_COMPONENTS; i++) {
                current_energy[i] += m[i];
        }
}
```

Figure 3.7: Monitoring Functions

3.3 DESCRIPTION OF THERMAL MODELS

In the following, prominent features of the thermal models which are integrated into the host-compiled simulation are described. They are used to generate transient and steady-state thermal profiles, and evaluate speed/accuracy tradeoffs in the process.

3.3.1 HotSpot

HotSpot [14] dynamically builds a representative RC model of the chip based on a given floorplan, and it computes the temperature profile of the chip over a sequence of time stamps (called sampling periods) based on a given trace of power dissipation values. HotSpot requires a structurally accurate power trace over all components in the floorplan

at the chosen sampling period, which is generated using the extended back-annotation approach described in the previous section. HotSpot provides two modes for temperature estimation: using block-based or grid-based models. The block model is chosen for computation of transient temperature profiles of a chip due to its advantage of higher speed during design space exploration.

HotSpot accepts a structurally accurate floorplan, representing each component by its coordinates in X-Y direction. This floorplan can be generated by 'hotfloorplan', one of the utilities provided by HotSpot. Input arguments to this utility are 1) area in mm^2 for each component, 2) desired aspect ratio for each component and 3) average power consumption of floorplan component.

3.3.2 Discrete Time Temperature Evaluation Model

DTTEM [8] [15] uses similar concepts as HotSpot for temperature estimation. This model requires conductance (G) and capacitance (C) matrices, which are characteristic of a chip floorplan. To maintain consistency between DTTEM and HotSpot thermal models, without loss of generality, RC representations from HotSpot are extracted and used by MATLAB to generate the parameters required by DTTEM. Any assumption or approximations made during generation of RC models in HotSpot are thus retained by DTTEM, making the comparisons easier.

The primary difference between DTTEM and HotSpot is the temperature evaluation mechanism. DTTEM assumes that, with sampling of power values at small and constant time intervals, transient temperature evaluation can be discretized. This assumption leads to a simplified solution of the basic first-order heat transfer differential equation, which is integrated into this flow to achieve faster temperature estimation. The solution to the differential equation is given as [8] [15]:

$\mathbf{T}[k+1] = e^{-\mathbf{A}\,\Delta t}\,\mathbf{T}[k] + \mathbf{A}^{-1}(\mathbf{I} - e^{-\mathbf{A}\,\Delta t})\,\mathbf{B}\,\mathbf{P}[k]$,

where $\mathbf{A} = \mathbf{C}^{-1}\mathbf{G}$, $\mathbf{B} = \mathbf{C}^{-1}$ and Δt is the sampling time interval.

It can be observed that the temperature estimation is a linear function of the current temperature of the nodes in the circuit and the power dissipation in the time interval. Moreover, the exponential term $e^{-\mathbf{A}\,\Delta t}$ can be pre calculated for a chosen sampling interval. Temperature for the next time interval can be calculated by considering two matrix multiplications and one matrix addition, reducing the evaluation time when compared to numerical approaches.

DTTEM can be used to estimate steady-state temperatures in a chip as well. The steady-state temperature reflects the thermal profile of the chip when the chip and environment are in equilibrium, i.e. when the benchmark is run repeatedly over and over on the same chip. Across such multiple iterations, power consumption in a particular interval can be calculated as average power consumption of the entire benchmark. Therefore, Steady-state power consumption is noted as $\mathbf{P}_{\mathbf{average}}$. Solving the above equation with $\mathbf{T}[k] = \mathbf{T}[k+1]$ for all values of $\mathbf{k}$ under steady-state assumptions yields

$\mathbf{T}[k] = e^{-\mathbf{A}\,\Delta t}\,\mathbf{T}[k] + \mathbf{A}^{-1}(\mathbf{I} - e^{-\mathbf{A}\,\Delta t})\,\mathbf{B}\,\mathbf{P}_{\mathbf{average}}$,

$\Rightarrow (\mathbf{I} - e^{-\mathbf{A}\,\Delta t})\,\mathbf{T}[k] = \mathbf{A}^{-1}(\mathbf{I} - e^{-\mathbf{A}\,\Delta t})\,\mathbf{B}\,\mathbf{P}_{\mathbf{average}}$,

$\Rightarrow \mathbf{T}_{\mathbf{ss}} = \mathbf{A}^{-1}\,\mathbf{B}\,\mathbf{P}_{\mathbf{average}}$,

$\Rightarrow \mathbf{Tss} = \mathbf{G}^{-1}\,\mathbf{C}\,\mathbf{C}^{-1}\,\mathbf{P}_{\mathbf{average}}$,

$\Rightarrow \mathbf{Tss} = \mathbf{G}^{-1}\,\mathbf{P}_{\mathbf{average}}$

$\mathbf{T}_{\mathbf{SS}} = \mathbf{G}^{-1}\mathrm{P}_{\mathrm{average}}$,

Figure 3.8 shows how both these thermal models are implemented in FastSpot setup.

```
//The function calculate_transient_temperature() captures model specific
operations.
//For the two models considered in our approach,

-------------------------------------------------------------------------------
//HotSpot
void calculate_temperature(double *Power_values){
        create_hotspot_pipe(estimate_steady_state);
        write_power_array(Power_values);
}
-------------------------------------------------------------------------------
//DTTEM
//NODES represent the number of junctions in RC  model of the floorplan
double T_coefficient[NODES][NODES];
double P_coefficient[NODES][NODES];
double P_current[NODES];
double T_current[NODES];
double T_next[NODES];
int     estimate_steady_state;

void calculate_temperature(double *Power_values){
        P_current = preprocess_power_values(Power_values);
        for (i = 0; i < NODES; i++){
                mmult = 0.0;
                for (j = 0; j < NODES; j++){
                        mmult += T_coefficient[i][j]*T_current[j] +
                                P_coefficient[i][j]*P_current[j];
                }
                T_next[i] = mmult;
                T_current[i] = T_next[i];
        }
        print_temperature_array();
        if (estimate_steady_state == 1)
                generate_steady_state (Power_values);
}
```

Figure 3.8: Thermal Model Implementations

3.4 INTEGRATION OF THERMAL MODELS

Figure 3.8 represents two implementations of the thermal evaluation function for HotSpot and DTTEM, respectively. For a HotSpot-based thermal model, power values are then written to a Unix pipe (function *write_power_array()*) created at the start of execution of the host-compiled model. HotSpot setup has been modified to accept data streamed through a pipe and interpret the written power values to calculate the tempera-

ture. This process, repeated every sampling period, generates a transient thermal trace of the original application.

For a DTTEM-based thermal model, the evaluation function implements the necessary matrix operations to generate the transient temperature at the end of current sampling period. The coefficient matrices are obtained by dumping parameters of the RC network constructed by HotSpot. Other matrix related computations are done in MATLAB, and a header file defining the coefficients (matrices *T_coefficient[][]* and *P_coefficient[][]*) is generated. This is a one-time procedure for a given floorplan and consumes a small amount of time.

After the temperature computation, *T_next[]* is fed to the output file/screen and *T_current[]* is populated with *T_next[]* for the next iteration. The function *preprocess_power_values()* rearranges and populates other nodes of the RC model for which *power_values[]* are not obtained in the host-compiled model. It should be noted that the number of nodes in the RC equivalent of the floorplan will be higher than the number of components, but the nodes acting as variable power sources (assuming point power sources at the center of the component) will be the same as the components.

The thermal model header file defines a variable *estimate_steady_state*, which can be used by the functional description of the model to evaluate steady-state temperature. In the case of HotSpot, it simply adds an argument to the HotSpot run command for steady-state estimations using the block-based model. For DTTEM, the variable enables steady-state thermal evaluation (function *generate_steady_state()*) using the formula noted earlier.

Back-annotated function calls are an overhead to the original application binary and ultimately result in a slowdown compared to pure functional execution of the appli-

cation. Therefore, the primary objective is to keep the function definitions simple and as small as possible. As described above, the functions do not contain many branch conditions (e.g. if statements) and the operations are mainly additions/multiplications. This reflects an objective to minimize overhead. Another issue in the HotSpot-based thermal model is a very high data transfer rate between the host-compiled model and HotSpot. The size of each data transfer depends on the number of floorplan components of the chip in consideration. Moreover, latency of the complete execution of the application and the sampling period dictates the number of such data transfers. Usage of files or print statements may slow down the thermal trace generation process considerably. Instead, FastSpot uses pipe calls, such as popen/pwrite in order to minimize impacts of the operating system. A DTTEM-based approach requires no such transfers, and such problems are not encountered in a DTTEM-based thermal estimation process.

Chapter 4: Reference Methodology

The reference flow developed for comparison is similar to the one developed in [3]. The methodology uses the uADL reference ISS from Freescale [4] to produce latency, energy and temperature traces. The front end in the reference flow is very straightforward, where a cross-compiled binary is executed on the ISS and an output trace is obtained. Using similar scripts as in the back-annotation flow, the methodology identifies latency information and access statistics for different sampling intervals. For each sampling interval, McPAT is invoked to obtain power values, and the dynamic power trace is fed into HotSpot for temperature trace generation. The floorplan is generated using hotfloorplan utility, included in HotSpot tool package. It is to be noted that floorplan generation process is a one-time task for a specific architecture. Figure 4.9 shows the steps involved in temperature trace generation using reference methodology.

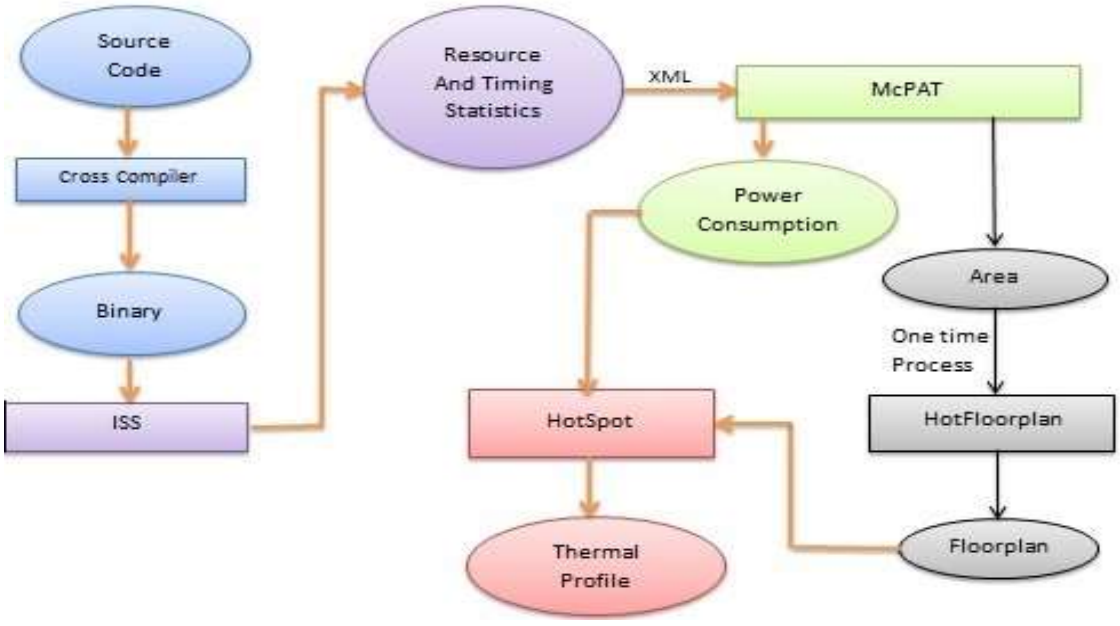

Figure 4.9: Reference Methodology

This flow is highly influenced by the dynamic instruction count of the benchmark. For example, assume that a benchmark has 1 million dynamic instructions with CPI (Cycles per instruction) of 1. The number of cycles required to execute the benchmark on ISS will be 1 million. If the power estimation interval is 10,000 cycles, power estimator has to work 100 times (= $10^6/10^4$). The higher the dynamic instruction count, more time will be spent in power estimations alone. The run time of the reference methodology is a strong function of speed of power estimators and dynamic instruction counts.

As seen later, the majority of the run time is spent in power calculations by McPAT due to periodic and repetitive calculation of power at the granularity of the sampling interval.

Chapter 5: Experiments and Results

In this Chapter, reference methodology and FastSpot are compared against each other and speed/accuracy tradeoffs are evaluated.

5.1 Evaluation Setup

FastSpot integrates the thermal estimation approach into the host-compiled back-annotation framework from [2]. The FastSpot tool is available for download at [13]. To demonstrate the feasibility of the proposed approach and to evaluate the tradeoffs, the back-annotation flow is applied to a simplified e200 Z6 (single issue) and e200 Z4 (dual issue) PowerPC like architecture. Reference models for the architecture are provided by Freescale through their uADL [4] framework. The chosen reference architectures can be classified as 32-bit processors having static branch predictors and no MMU, cache or FP units. Modeling of such components is considered future work. Although the uADL models do not include MMUs or caches, area estimates of McPAT and the floorplan generated by hotfloorplan model these components. The primary reason is to study a more generic floorplan from a temperature variation perspective. Accesses to these otherwise unavailable components are always zero. Hence, they do not contribute to dynamic power.

The reference methodology and FastSpot are compared on the basis of simulation throughput, run time and relative accuracy of results. The benchmarks are chosen from the MiBench suite [6] [11] namely, ADPCM (Telecom), CRC32 (Telecom) and SHA (Security) - for comparison due to the highly pervasive nature of mobile computing. Benchmarks are compiled using gcc with O2 optimizations. Small input datasets are used and file I/O operations are converted into array operations. Dynamic instruction counts of

benchmarks are obtained by executing them on the cycle-accurate reference ISS, and are listed in Table 5.2.

Benchmark	Suite	Instruction count	Unique BB pairs
ADPCM	Telecom	36,667,034	53
CRC32	Telecom	12,319,886	7
SHA	Security	14,698,630	88

Table 5.2: Benchmark Details

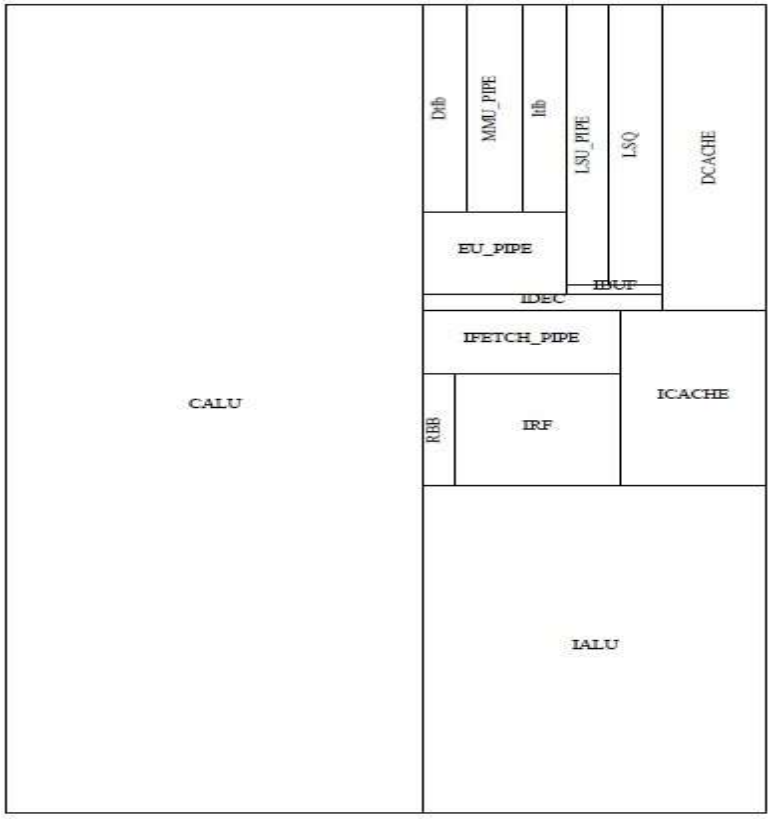

Figure 5.10: Z4 Floorplan

A custom application is also written in order to stress different architecture components and thus study variations in hotspot generation. The simulations are performed on Z6 and Z4 like architectures with a floorplan area (for 90 nm process technology) of 4.77 mm^2 and 6.85 mm^2, respectively. Figure 5.10 and 5.11 show placement of components with respect to each other in the generated floorplan for Z4 and Z6 architecture, respectively.

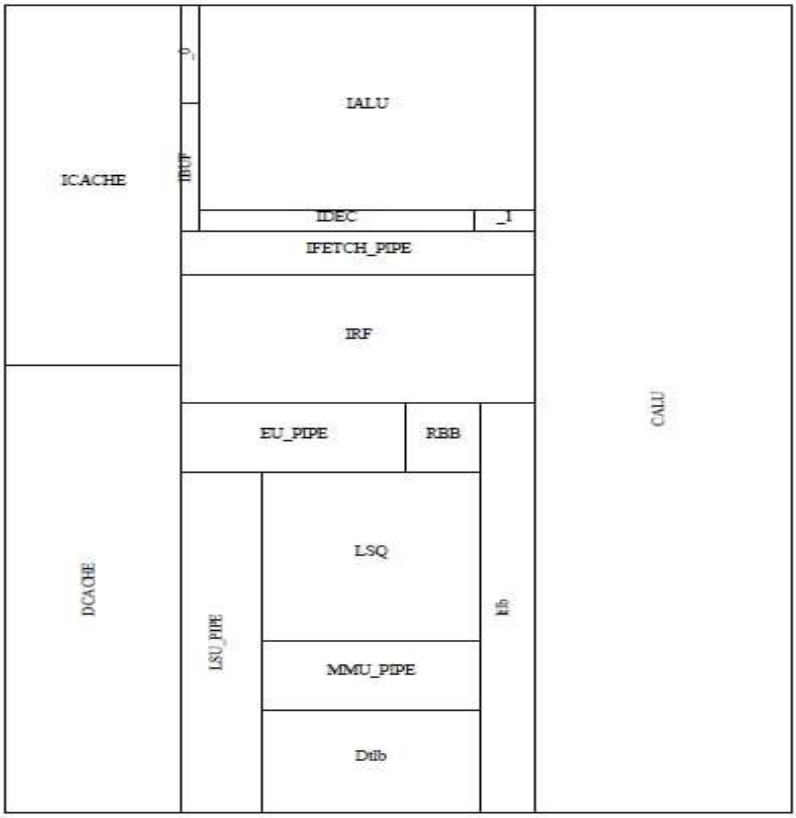

Figure 5.11: Z6 Floorplan

The operating frequency is assumed to be 500 MHz and ambient temperature to be 318.15K. A sampling period of 10k cycles (unless stated otherwise) has been chosen for a good tradeoff between precision and overhead in HotSpot as suggested in [14]. The

methodology uses McPAT version 0.8 and the block-based model (unless stated otherwise) of HotSpot version 5.02.

5.2 SPEED COMPARISON

Two metrics are considered for speed evaluations: (1) total run time and (2) simulation throughput. The reason is to quantify combined characterization and simulation time in FastSpot. For the reference flow, run time includes ISS execution time, total McPAT run time and HotSpot (HS) run time. For the FastSpot (FS) flow, the contributors to the run time are characterization time and execution time of the host-compiled model implementing the thermal model. For DTTEM-based FastSpot, thermal parameter generation time is also included in the calculation.

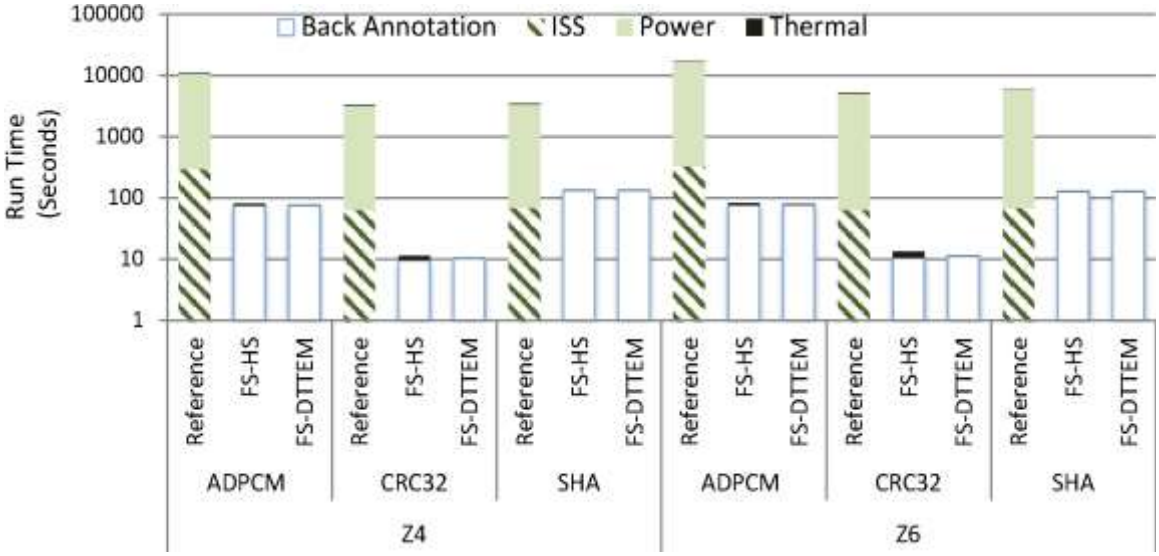

Figure 5.12: Run Time Comparison

Figure 5.12 shows the runtime breakdown of the reference flow and the two FS methodologies. The runtime improvements are similar in case of the Z4 and Z6 models. Overall, McPAT invocations contribute to a majority of the run time (ADPCM- 97.7%,

CRC32- 97.8%, SHA- 97.9%) in the reference flow. By contrast, the major bottleneck in the proposed methodology is in the back-annotation time. It can be seen that for execution of a single benchmark with small inputs, time spent in generation and extraction of thermal parameters is insignificant and the choice of thermal model (HotSpot or DTTEM) does not play a major role. When compared to the run time of the reference flow, the run time of the FS flow in case of the Z6 model is 214, 503 and 43 times lower for ADPCM, CRC32 and SHA, respectively. In case of the Z4 model, run time of the FS flow is 125, 280 and 25 times lower for ADPCM, CRC32 and SHA, respectively. Time spent in characterization of the blocks of a benchmark depends on its control flow graph complexity. If there are a large number of unique block pairs in an application with low dynamic instruction counts, there is less improvement in total run time. The time spent in thermal parameter generation is fairly constant across these two architectures.

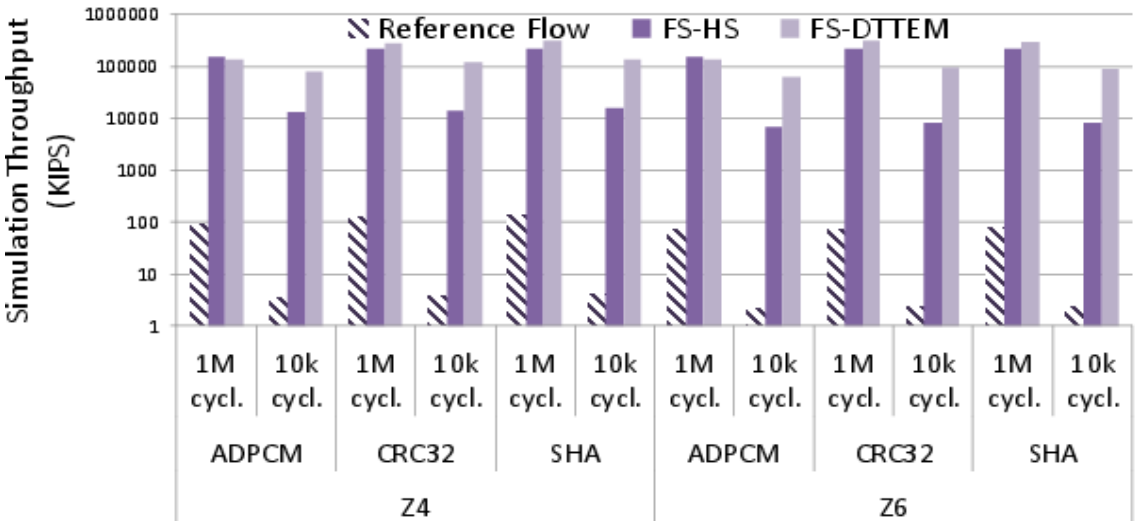

Figure 5.13: Simulation Throughput Comparison

The simulation throughput is calculated as the ratio of the dynamic instruction count of a benchmark to the execution time of the host-compiled model. Back-annotation

time is not accounted in the calculation of simulation throughput, as this is a one-time offline procedure for a given benchmark. After a benchmark has been characterized for all blocks, execution time of the annotated binary controls the simulation throughput, e.g. for repeated simulations over larger input data sets. Figure 5.13 shows variations in the simulation throughput between two different thermal-aware simulations with sampling periods of 1 million and 10,000 cycles. Note that simulation throughput increases considerably with an increase in the sampling period. Therefore, the sampling period can be adjusted for higher throughput but less accurate temperature profiles.

Benchmark	FS (DTTEM)	FS (HS-Block)	FS (HS-Grid)
ADPCM (Z4)	136 MIPS	115 MIPS	99.1 MIPS
CRC32 (Z4)	385 MIPS	246 MIPS	162 MIPS
SHA (Z4)	298 MIPS	207 MIPS	154 MIPS
ADPCM (Z6)	131 MIPS	108 MIPS	96.5 MIPS
CRC32 (Z6)	373 MIPS	176 MIPS	124 MIPS
SHA (Z6)	281 MIPS	191 MIPS	119 MIPS

Table 5.3: Steady-state Simulation Throughput

For a sampling period of 10,000 cycles, average simulation throughput of the proposed methodology with a HotSpot model across both processors is 11 MIPS, whereas the simulation throughput for the reference flow is just 3.1 KIPS, making the former around 3600 times faster. When the comparisons are done for the proposed approach with DTTEM integration, the simulation throughput is 98 MIPS (a speedup over the reference

flow of 32,000x). Overall, a DTTEM model achieves about a 9x speedup over a HotSpot based thermal evaluation.

It is to be noted that FastSpot with DTTEM does not achieve a speedup with respect to HotSpot in the steady-state (SS) estimation process. Table 5.3 summarizes SS throughput results. Simulation throughput is of the same order and does not improve significantly with a DTTEM-based model. The reason is that the steady-state temperature measurement is not affected much by the dynamic instruction count of the benchmarks, but transient temperature trace generation is. In the proposed FS methodology, the transient temperature generation is the primary bottleneck, and this thesis proposes using DTTEM to overcome the issue.

5.3 ACCURACY COMPARISON

In the following, relative accuracy of the back-annotation flow is compared to the reference flow in terms of latency and temperature. The absolute and percentage errors for latency match the results obtained in [2]. The average error in latency measurement is 1.81% for the set of three benchmarks, where the highest error is present for ADPCM (5.38%). Errors in latency measurements in the host-compiled simulation are the primary source of all temperature errors. Latency deviations get propagated into energy and temperature errors, since power and temperature are calculated from latency information. The loss in accuracy for host-compiled latency measurements is mainly because of not being able to exactly replicate pipeline states for a basic block when characterizing blocks only across immediate predecessors. In addition, as results will show, use of DTTEM leads to additional errors stemming from approximations in the temperature model.

For temperature estimation, there is not a single number to compare as the output is a trace over time. Two metrics are developed, (1) transient temperature error and (2) steady-state temperature error, for evaluating temperature accuracy of the proposed flow. Average values for these absolute errors are calculated using the following equations:

Average absolute transient temperature error

$= (\sum_{fp}\sum_{t} |\mathbf{T}(fp,t)_{back\text{-}annotation} - \mathbf{T}(fp,t)_{reference}|) / (Number\_Components * Trace\_Length)$

For all **fp** in floorplan components and for all **t** in Time points.

Average absolute steady-state temperature error

$= (\sum_{fp} |\mathbf{T_{ss}}(fp)_{back\text{-}annotation} - \mathbf{T_{ss}}(fp)_{reference}|) / Number\_Components$

For all **fp** in floorplan components.

To calculate the transient temperature errors, temperature traces of the reference flow and the back-annotation flows are compared for all floorplan components.

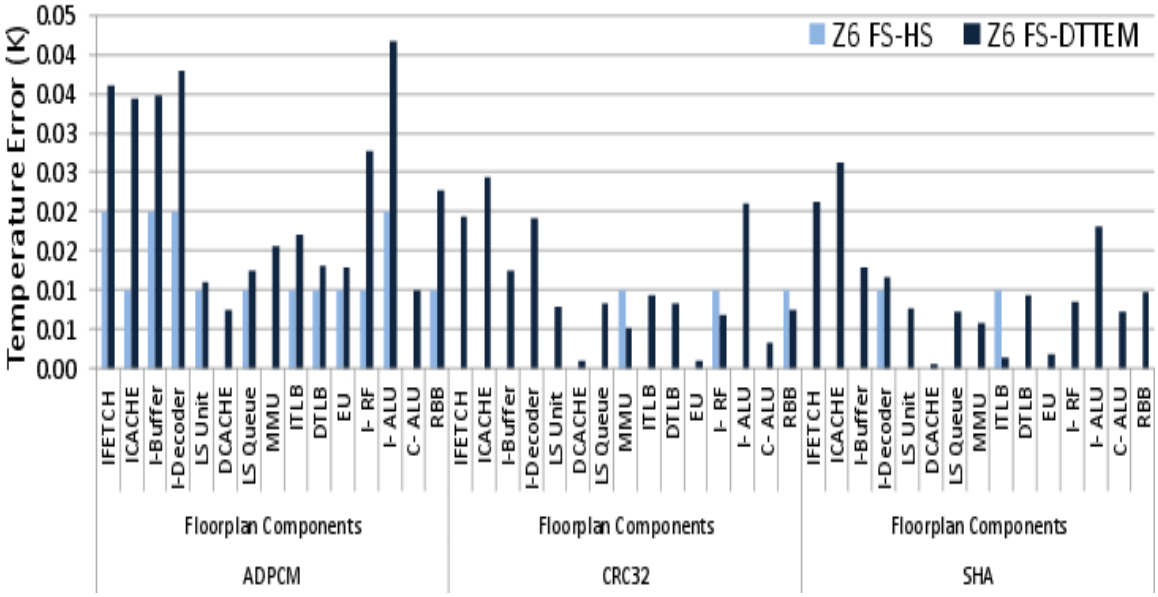

Figure 5.14: Steady-state Temperature Errors (Z6)

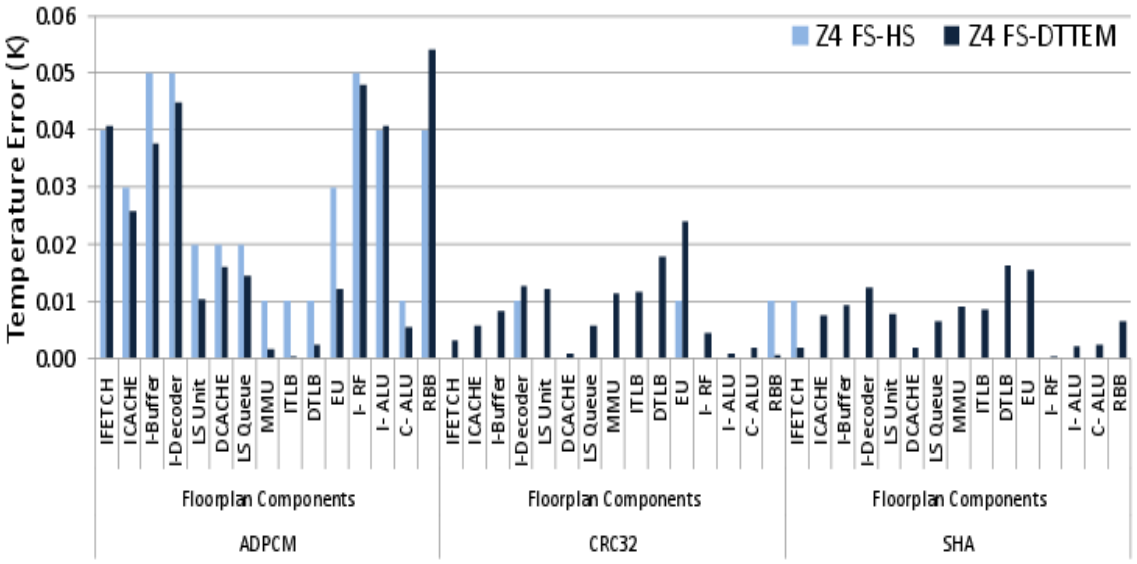

Figure 5.15: Steady-state Temperature Errors (Z4)

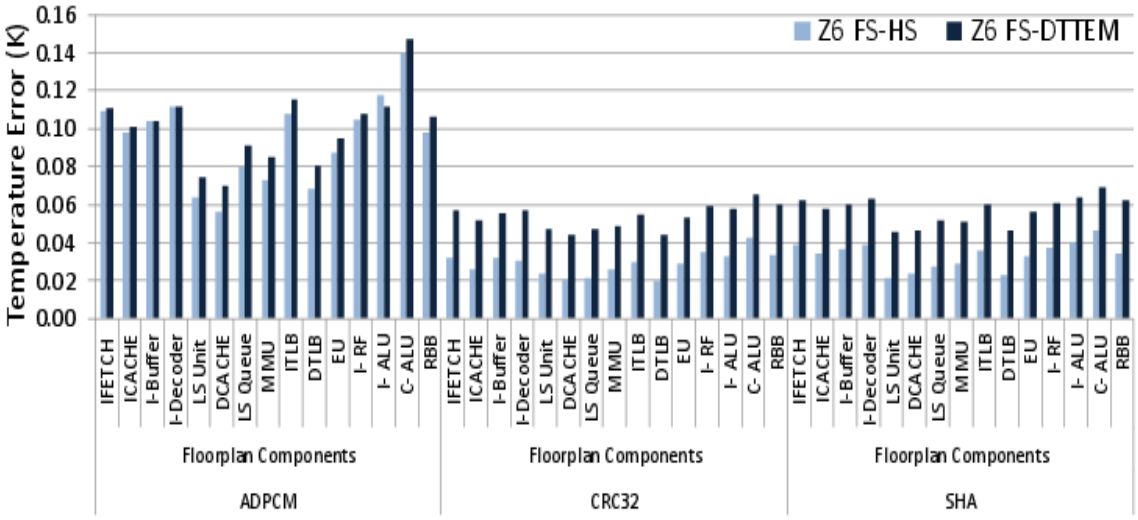

Figure 5.16: Transient Temperature Errors (Z6)

Figure 5.14 and 5.15 show errors in steady-state temperature measurement for Z6 and Z4 architecture, respectively. The measurement error for each floorplan component is calculated and absolute values are shown in the graph. Figure 5.16 and 5.17 show errors

in transient temperature measurement. The absolute difference between component temperatures is noted and averaged for all estimation intervals.

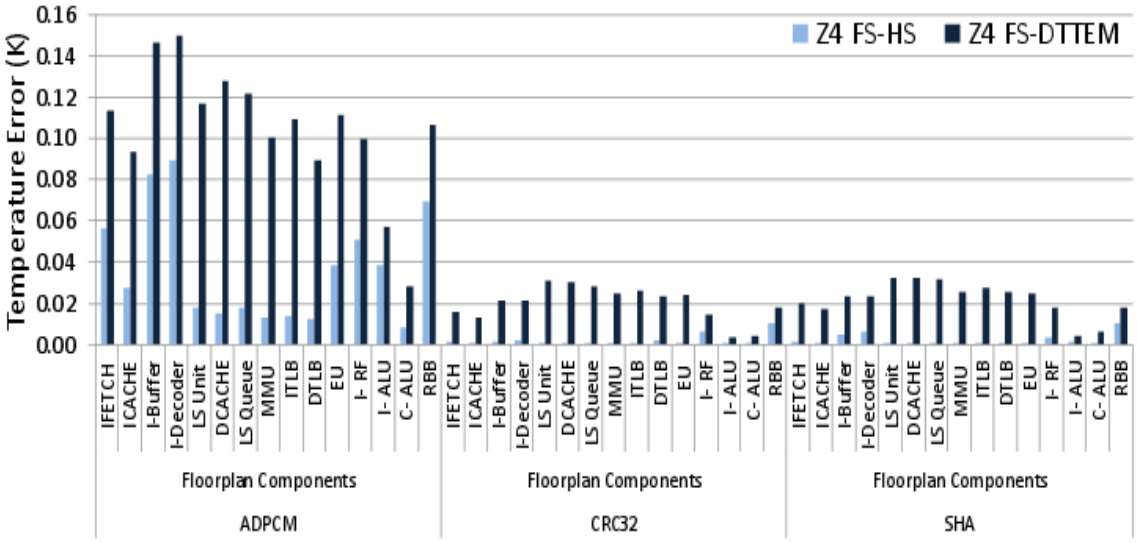

Figure 5.17: Transient Temperature Errors (Z4)

When HotSpot is used as the thermal model for the Z6 architecture, the maximum error is 0.14 K, whereas the maximum error in case of the DTTEM model is 0.15 K. Similarly, maximum errors in the Z4 case are 0.09 K and 0.14 K, respectively. The average error across all components of all benchmarks is 0.019 K higher than the HotSpot-based methodology in case of the DTTEM-based methodology for the Z6 processor. Similarly, the DTTEM-based methodology incurs a higher error (by 0.035 K) than the HotSpot-based implementation of FastSpot for the Z4 processor. To calculate steady-state temperature errors, HotSpot is configured to generate a temperature profile using the block-based model. HotSpot reports steady-state temperature for all the components of a floorplan, which are compared against the values obtained from DTTEM. This metric is useful in practical analysis for studying temperature variation and hotspots.

When HotSpot is used as the thermal model, the maximum error is 0.02 K for the Z6 processor and 0.05 K for the Z4 processor. The maximum errors in case of the DTTEM-based thermal model are 0.041 K and 0.054 K, respectively.

Table 5.4 summarizes the transient and steady-state temperature errors over all floorplan components for both thermal models. In case of the HotSpot based FastSpot, the average transient error is 0.05K for the Z6 and 0.014 K for the Z4 architecture. By contrast, in case of the DTTEM model, average transient errors are 0.07 K and 0.05 K, respectively. The average absolute steady-state errors in both the models are low and usage of DTTEM instead of HotSpot increases the average steady-state error only marginally for the Z6 model (0.014 K versus 0.0047 K for a 0.009 K increase). In case of the Z4 model, usage of DTTEM increases the average steady-state error from 0.011 K to 0.013 K for a 0.002 K increase.

Benchmark	**Transient**		**Steady-state**	
	FS-HS	FS-DTTEM	FS-HS	FS-DTTEM
ADPCM (Z4)	0.037 K	0.1 K	0.029 K	0.024 K
CRC32 (Z4)	0.0017 K	0.02 K	0.002 K	0.008 K
SHA (Z4)	0.002 K	0.022 K	0.0007 K	0.0072 K
ADPCM (Z6)	0.09 K	0.1 K	0.011 K	0.022 K
CRC32 (Z6)	0.029 K	0.053 K	0.002 K	0.01 K
SHA (Z6)	0.033 K	0.057 K	0.001 K	0.01 K

Table 5.4: Average Absolute Temperature Errors

Maximum transient errors in a single sampling point range between 0.53 K and 0.75 K for a HotSpot and 0.55 K to 0.88 K for a DTTEM-based methodology. The high-

est observed chip temperature is 320.2 K for the Z4 and 319.6 K for the Z6 processor. Transient errors are higher in case of DTTEM mainly because of the approximations related to discretization. The solution to the heat transfer equation is approximated in the DTTEM-based approach. For example, it cannot capture the secondary effects of current temperature values on the next set of temperatures.

Chapter 6: Hotspot Characterization

In this section, comprehensive analysis of steady-state thermal profile is presented between the reference flow and FastSpot using a grid-based model of HotSpot with a grid size of 64x64. The grid model of HotSpot divides the floorplan into layers of a grid and performs computation on small component cubes. Different benchmarks have different characteristics and may produce dissimilar temperature profiles when executed on a micro architecture. Thermal profile of all the experiments is shown here. ADPCM (Figure 6.18 and 6.19), CRC32 (Figure 6.20 and 6.21 and SHA (Figure 6.22 and 6.23) benchmarks have similar thermal profiles. These thermal profiles depict a hotspot over instruction fetch unit. A manual stressmark was also written to stress various architectural components. The stressmark focuses on Complex ALU operations (Mul/Div) and produces two hotspots at the instruction fetch block and the Result Broadcast Bus.

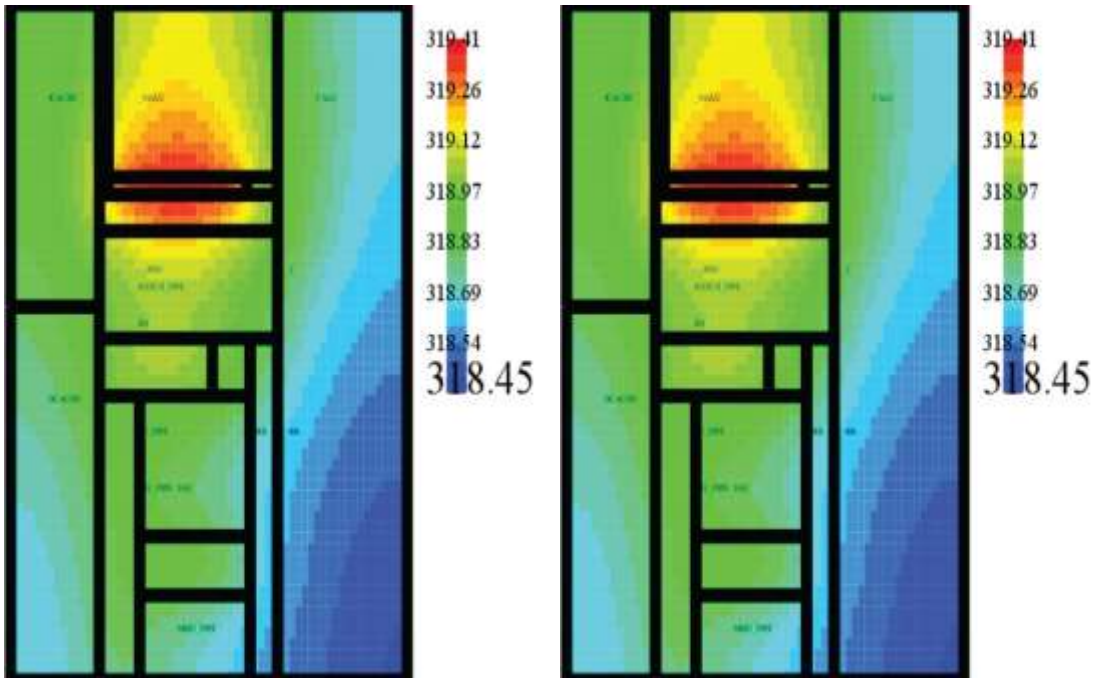

Figure 6.18: Steady-state Profile (Z6) for ADPCM (left- FastSpot, right-Reference)

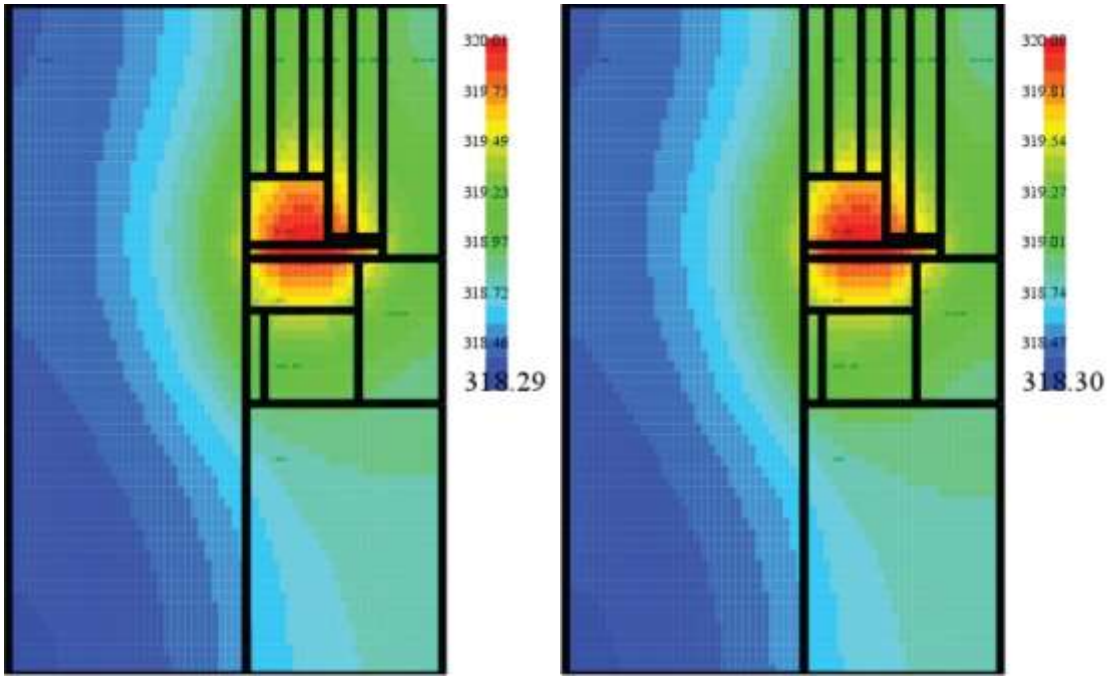

Figure 6.19: Steady-state Profile (Z4) for ADPCM (left- FastSpot, right-Reference)

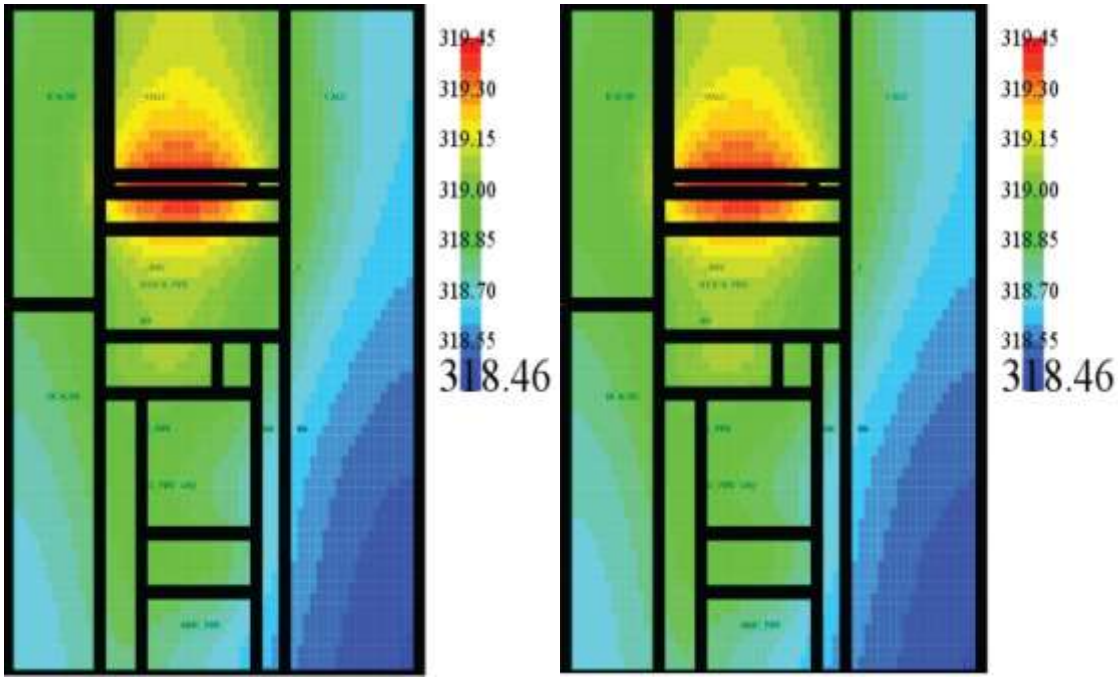

Figure 6.20: Steady-state Profile (Z6) for CRC32 (left- FastSpot, right-Reference)

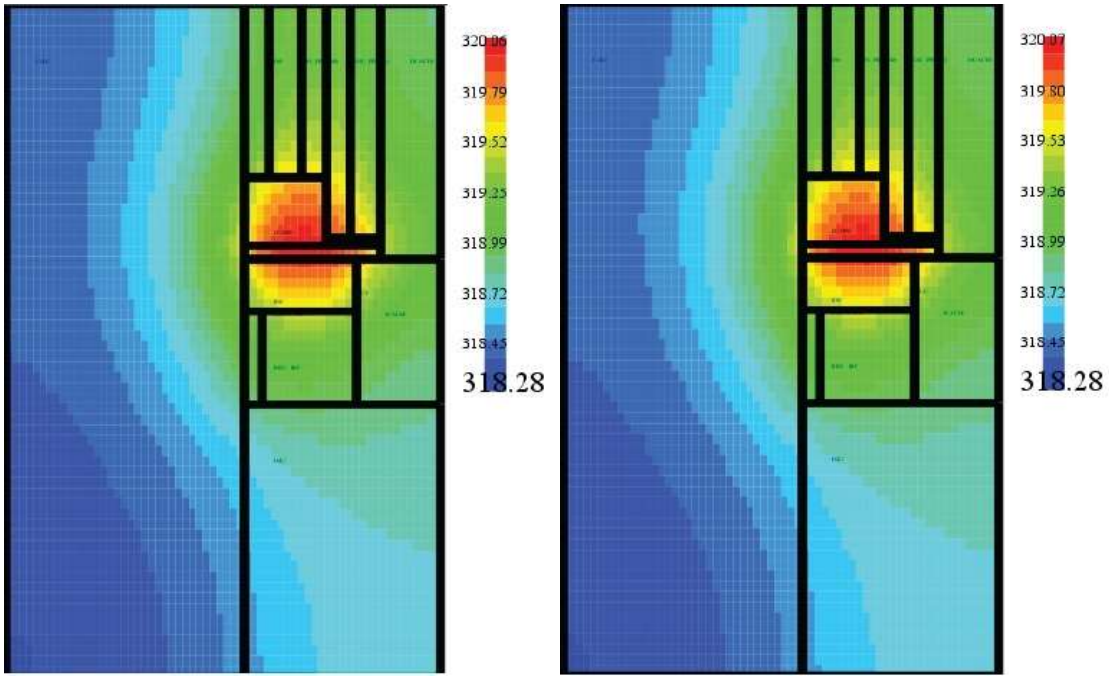

Figure 6.21: Steady-state Profile (Z4) for CRC32 (left- FastSpot, right-Reference)

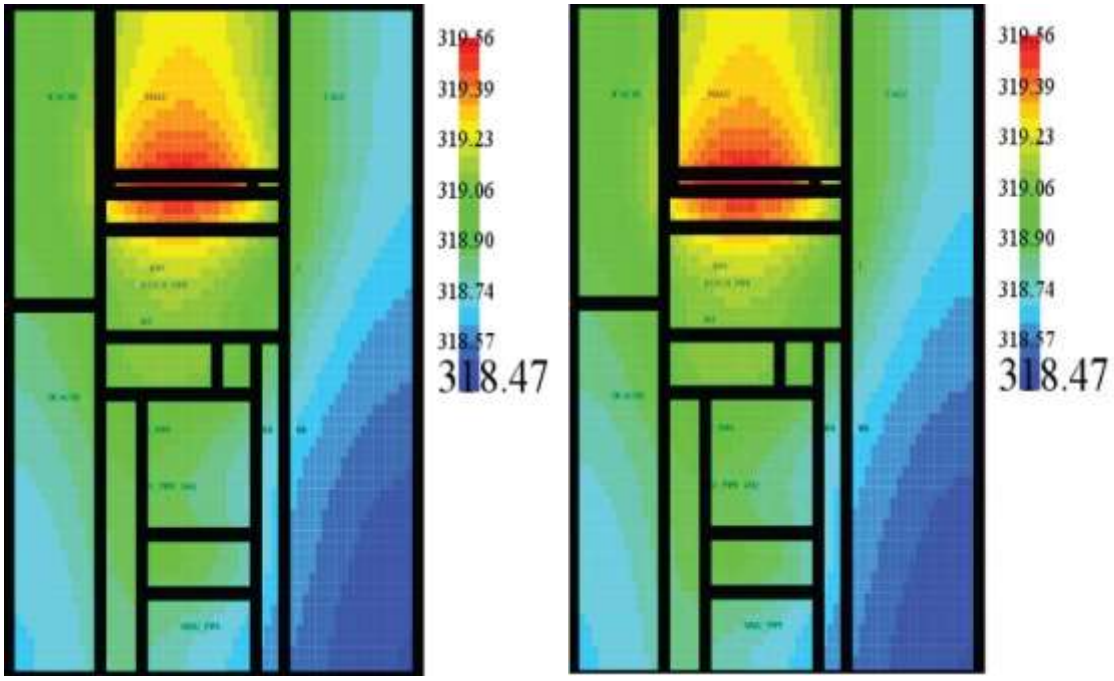

Figure 6.22: Steady-state Profile (Z6) for SHA (left- FastSpot, right-Reference)

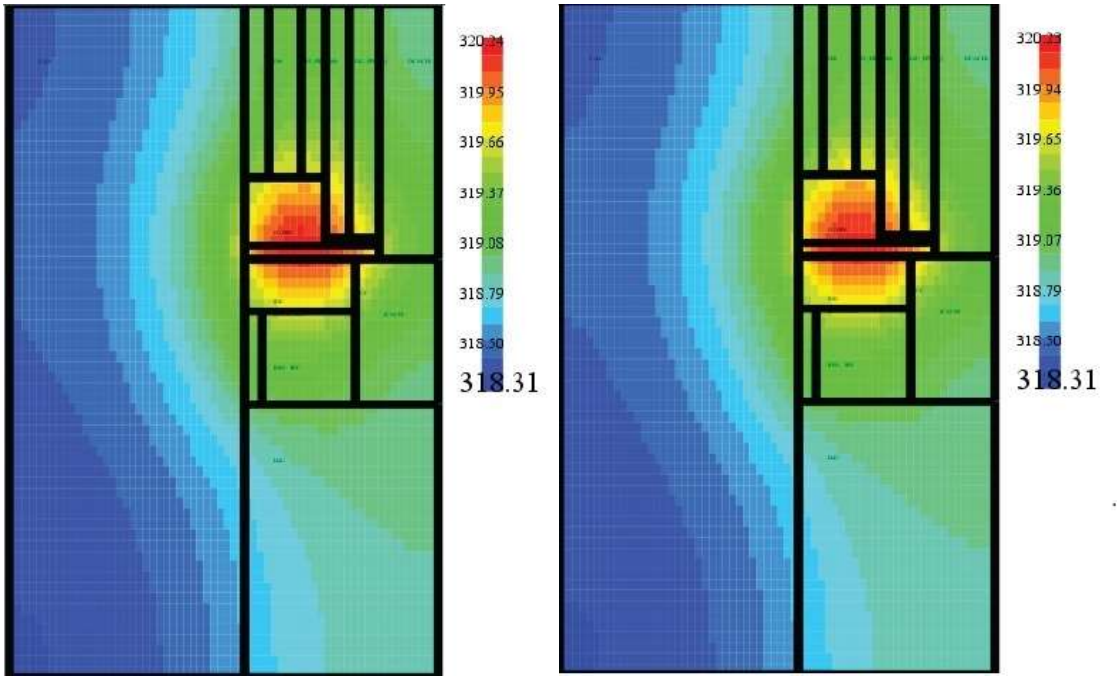

Figure 6.23: Steady-state Profile (Z4) for SHA (left- FastSpot, right-Reference)

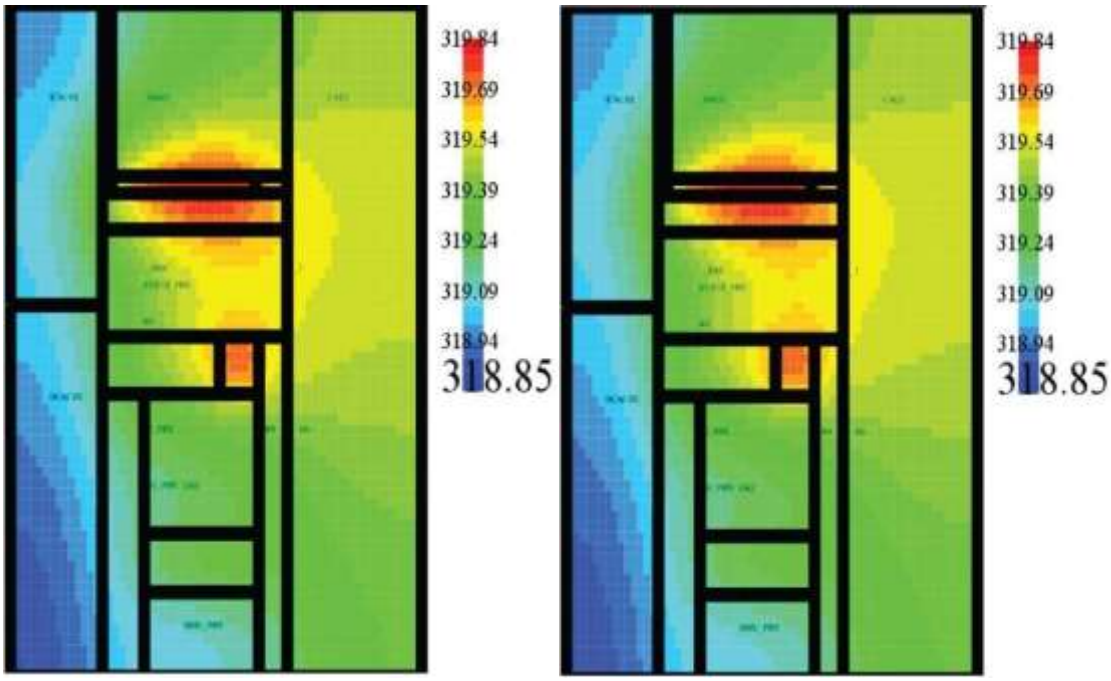

Figure 6.24: Steady-state Profile (Z6) for stressmark (left- FastSpot, right-Reference)

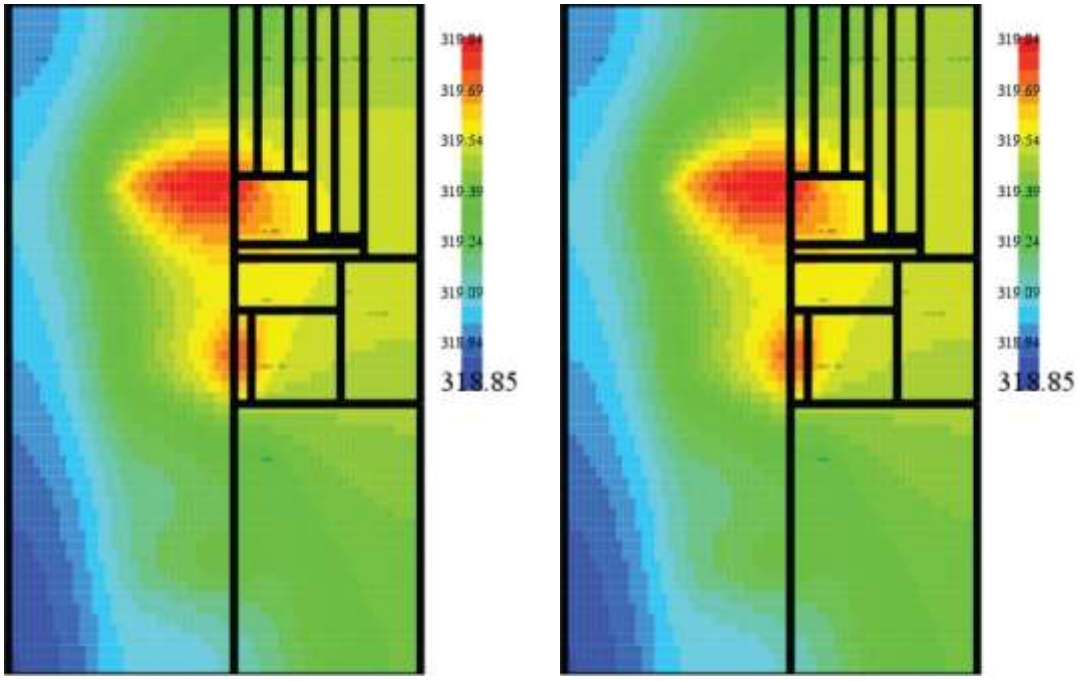

Figure 6.25: Steady-state Profile (Z4) for stressmark (left- FastSpot, right-Reference)

It can be observed visually that FastSpot flow accurately tracks high temperature hotspot and temperature variations across different benchmarks and that there are only minimal differences in steady-state temperature profiles. Furthermore, it does so at significantly reduced runtime (see Table 5.3).

In order to quantify thermal profile difference, error profiles are developed on 2-D space by subtracting steady-state thermal profiles of FastSpot and reference. Figure 6.26 and 6.27 provides these profiles for ADPCM benchmark for Z4 and Z6 architecture, respectively. The error profiles for other benchmarks have relatively lower error variations. CRC32 benchmark error profiles are shown in figure 6.28 and 6.29, while SHA benchmark error profiles are shows in figure 6.30 and 6.31. These graphs are generated using mesh graph capability of MATLAB tool.

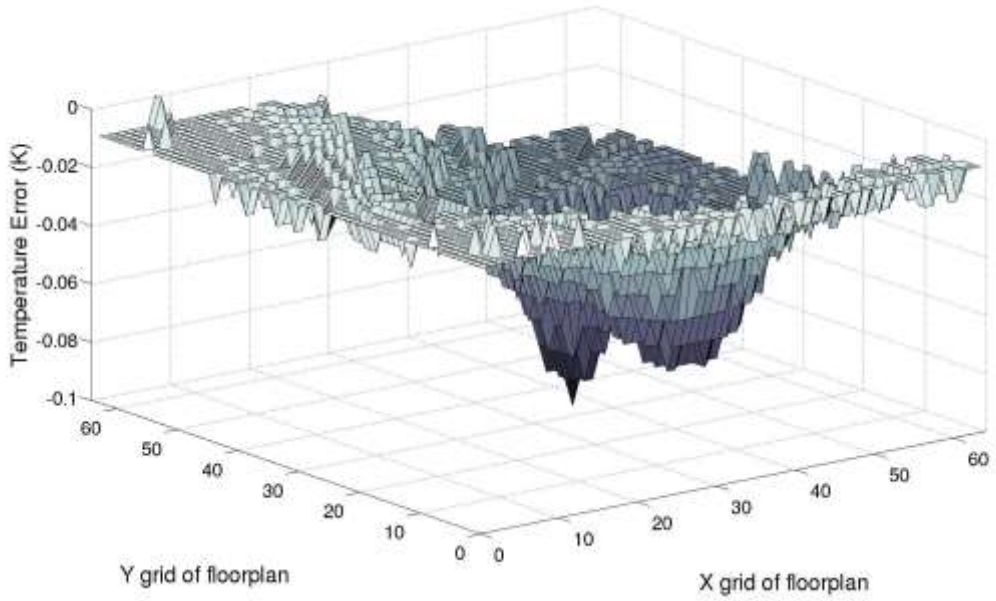

Figure 6.26: Steady-state Error Profile for ADPCM (Z4)

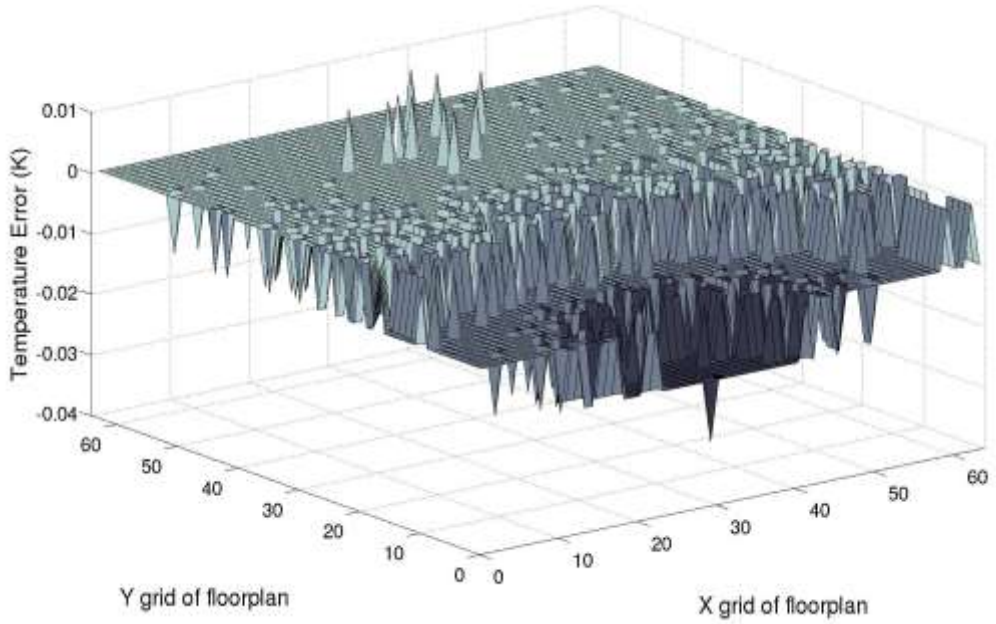

Figure 6.27: Steady-state Error Profile for ADPCM (Z6)

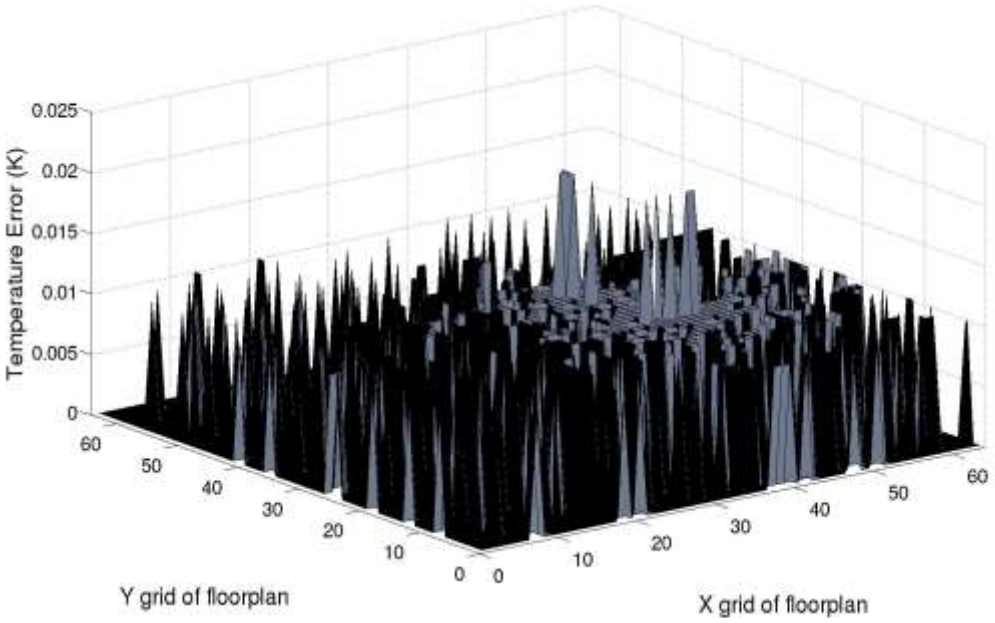

Figure 6.28: Steady-state Error Profile for SHA (Z4)

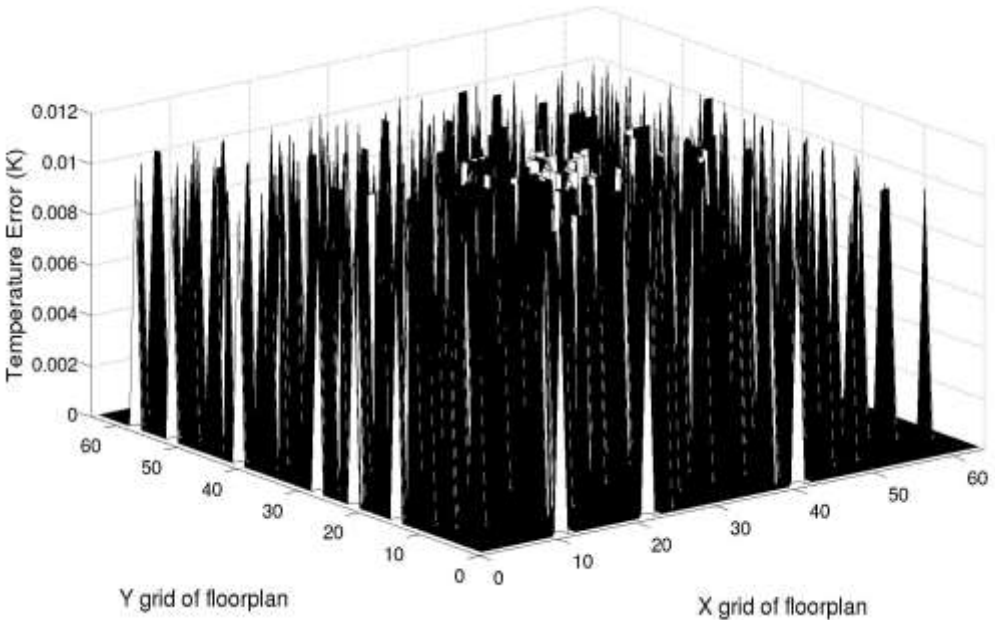

Figure 6.29: Steady-state error profile for SHA (Z6)

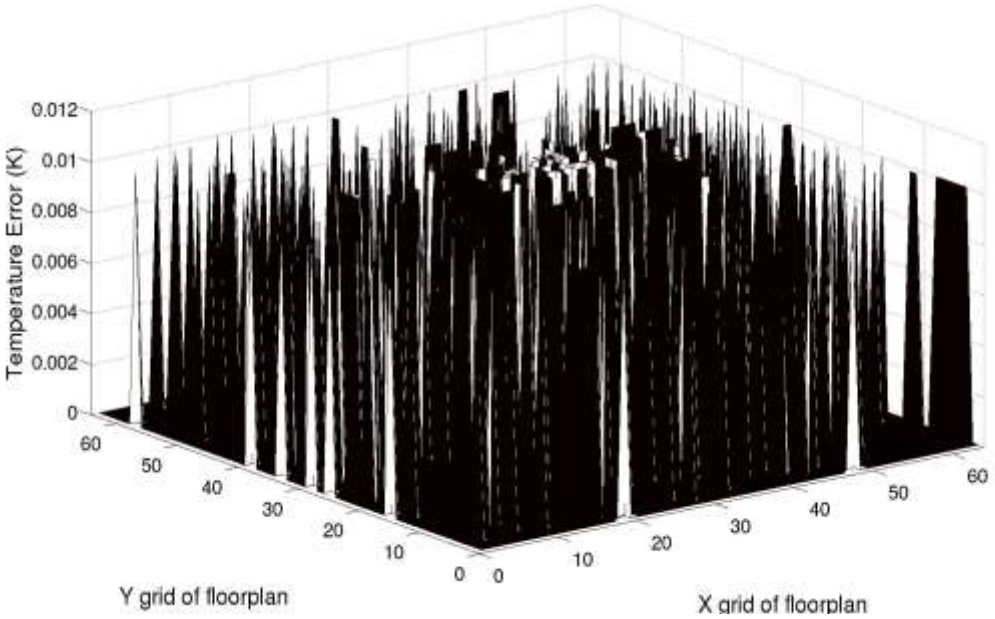

Figure 6.30: Steady-state Error Profile for CRC32 (Z4)

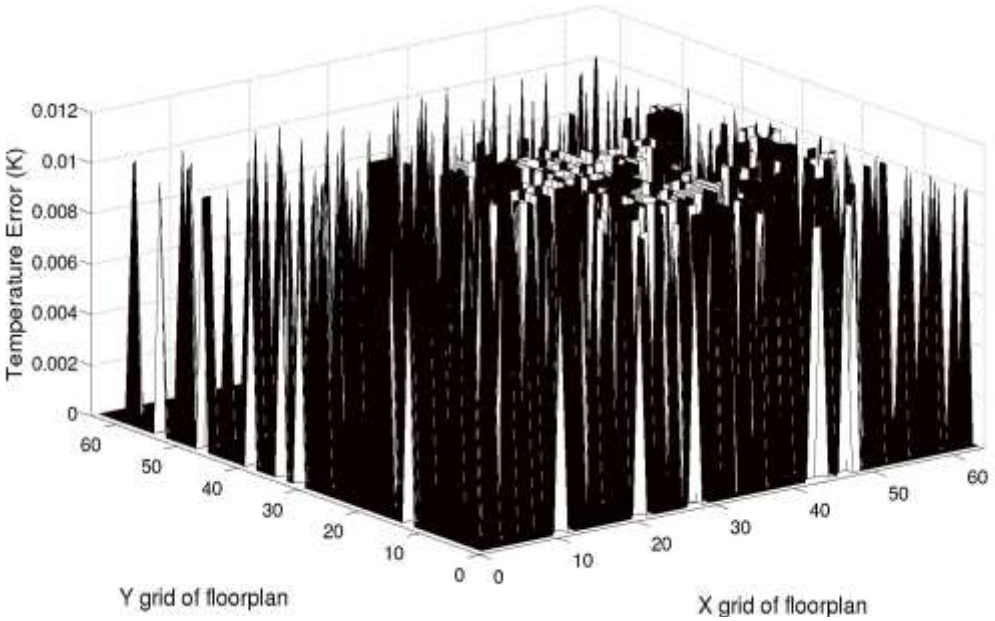

Figure 6.31: Steady-state Error Profile for CRC32 (Z6)

Chapter 7: Summary, Conclusions and Future Work

In this thesis, a novel simulation methodology for fast and accurate thermal estimations is proposed. This method uses host-compiled simulation instead of conventional ISS-based models. During a characterization phase, the source code of an application is annotated at block granularity with estimates, such as latency and energy consumption. Upon execution, the annotated binary creates a power trace and feeds the same into the thermal model. The thermal model operates on this trace and a floorplan to provide a final thermal trace and profile. The flexibility and integration capability of the methodology is demonstrated by integrating two different thermal models, HotSpot and DTTEM.

Using DTTEM as the thermal model, a simulation throughput of 98 MIPS is achieved, which represents a 32,000x speedup compared to a traditional flow for combined latency, power and temperature characterization. The error in measurement of temperature is very low. The reported errors are 0.06 K and 0.014 K average absolute errors in transient and steady-state temperature generation, respectively. The DTTEM-based FastSpot methodology has major speed benefits, which come at the cost of increased measurement error when compared to the HotSpot-based implementation (which is around 9 times slower). The methodology incurs a 0.02 K increase in transient temperature error and a 0.009 K increase in steady-state temperature error for the Z6 model. Similarly, transient and steady-state measurement errors increase by 0.035 K and 0.002 K, respectively for the Z4 model. The sampling period can play an instrumental role in reducing the effects of approximations in the DTTEM model. Results demonstrate the consistency and accuracy of FastSpot flow for temperature measurement when compared to the reference flow, which indicates its viability for generation of thermal profiles during early design space exploration.

The FastSpot methodology spends a majority of the execution time in running HotSpot for temperature trace generation. FastSpot methodology therefore integrates a faster and approximate thermal model (DTTEM) in the approach. Further extensions could include investigating implications of other fast thermal estimation methods [7] [10] [12] [16]. Evaluation of simulation bottlenecks can be beneficial for further optimizations. For example, fine-tuning the annotated code being added to the source code can reduce overhead and improve simulation throughput. Finally, there is a scope of improvement in accuracy and scope, e.g. through modeling of dynamic components, such as MMUs, branch predictors or caches that are currently not considered.

References

[1] Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In International Symposium on Computer Architecture (ISCA), 2000.

[2] S. Chakravarty, Z. Zhao, and A. Gerstlauer. Automated, retargetable back-annotation for host-compiled performance and power modeling. In International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2013.

[3] S. Eratne et al. Reducing thermal hotspots in multi-core processors using dynamic core scheduling. In International Conference on Computer Design (CDES), 2011.

[4] Freescale. ADL release 2.0.0. [Online] http://opensource.freescale.com/fsl-oss-projects.

[5] A. Gerstlauer. Host-compiled simulation of multi-core platforms. In IEEE International Symposium on Rapid System Prototyping (RSP), 2010.

[6] M. Guthaus et al. Mibench: A free, commercially representative embedded benchmark suite. In IEEE International Workshop on Workload Characterization, 2001.

[7] Y. Han, I. Koren, and C. M. Krishna. TILTS: A fast architectural-level transient thermal simulation method. Journal of Low Power Electronics, 3(1):13–21, 2007.

[8] A. Krum. Thermal Management. In F. Kreith, The CRC Handbook of Thermal Engineering. CRC Press, 2000.

[9] S. Li et al. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In IEEE/ACM International Symposium on Microarchitecture (MICRO), 2009.

[10] P. Liu et al. Fast thermal simulation for architecture level dynamic thermal management. In IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2005.

[11] MiBench version 1.0. [Online] http://www.eecs.umich.edu/mibench/source.html.

[12] J. Nayfach-Battilana and J. Renau. SOI, interconnect, package, and mainboard thermal characterization. In International Symposium on Low Power Electronics and Design (ISLPED), 2009.

[13] Retargetable Back-Annotator (RBA), version 1.1.

[Online]. http://www.ece.utexas.edu/~gerstl/releases/.

[14] K. Skadron et al. Temperature-aware microarchitecture. In International Symposium on Computer Architecture (ISCA), 2003.

[15] L. Thiele, L. Schor, H. Yang, and I. Bacivarov. Thermal-aware system analysis and software synthesis for embedded multi-processors. In Design Automation Conference (DAC), 2011.

[16] Y. Yang et al. ISAC: Integrated space-and-time-adaptive chip-package thermal analysis. IEEE Transactions on CAD of Integrated Circuits and Systems, 26(1):86–99, 2007.

[17] A. Ziabari et al. Fast thermal simulators for architecture level integrated circuit design. In IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2011.

[18] R. R. Schaller. Moore's law: past, present and future, IEEE Spectrum, vol.34, no.6, pp.52-59, 1997.

[19] T. Kemper, Y. Zhang, Z. Bian, and A. Shakouri. Ultrafast Temperature Profile Calculation in IC chips, Proc. of 12th International Workshop on Thermal investigations of ICs (THERMINIC), France, pp.133-137, 2006.

[20] Standard Performance Evaluation Corporation (SPEC) 2006 benchmark suite. [Online]. Available http://www.spec.org/cpu2006/.

[21] Synopsys Corporation. Powermill data sheet, 1999.

[22] T. Austin, E. Larson, D. Ernst. SimpleScalar: an infrastructure for computer system modeling, Computer, vol.35, no.2, pp.59-67, 2002.

[23] A. Kahng, B. Li, L.-S. Peh, and K. Amadi. ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration, in Design Automation and Test in Europe (DATE), 2009.

[24] San-Shan Hung, Chang Hsing-Cheng, You-Lun Chen, Ming-Hua Liu. High Efficiency Liquid-type Cooling and Packaging Technology for a Miscellaneous Application of Microprocessors, International Conference on Electronic Packaging Technology (ICEPT), pp.1-4, 2007.

[25] R. C. Jaeger, J. S. Goodling, M. E. Baginski, C. D. Ellis, N. V. Williamson, R. M. O'Barr. High heat flux cooling for silicon hybrid multichip packaging, IEEE Transactions on Components, Hybrids, and Manufacturing Technology, vol.12, no.4, pp.772-779, 1989.

[26] R. Jayaseelan, T. Mitra. A hybrid local-global approach for multi-core thermal management, Computer-Aided Design - Digest of Technical Papers, pp.314-320, Nov. 2009.

[27] Junyoung Park, H. M. Ustun, J. A. Abraham. Run-time Prediction of the Optimal Performance Point in DVS-based Dynamic Thermal Management, 25th International Conference on VLSI Design (VLSID), pp.155-160, 2012.

[28] A. Joshi, Aashish Phansalkar, L. Eeckhout, L. K. John. Measuring benchmark similarity using inherent program characteristics, IEEE Transactions on Computers, vol.55, no.6, pp.769-782, 2006.

[29] J. Schnerr, O. Bringmann, A. Viehl, and W. Rosenstiel. High performance timing simulation of embedded software, in Design Automation Conference (DAC), CA, 2008.

[30] R. D'omer. Transaction level modeling of computation, Center for Embedded Computer Systems, University of California-Irvine, Tech. Rep. CECS-06-11, 2006.

[31] J. Nayfach and J. Renau. SOI, Interconnect, Package, and Mainboard Thermal Characterization, International Symposium on Low Electronics and Design (ISLPED), pp.327-330, 2009.

[32] J. H. Park, S. Shin, J. Christofferson, A. Shakouri and S. M. Kang. Experimental Validation of the Power Blurring Method, Proceedings of 26th Semi-Therm, Santa Clara, pp.240-244, 2010.

[33] ANSYS. [Online]. Available: http://www.ansys.com.

[34] International Technology Roadmap for Semiconductors. [Online]. Available: http://public.-itrs.net.

[35] Mentor Graphics, 1999.

[36] S. J E Wilton, N. P. Jouppi. CACTI: an enhanced cache access and cycle time model, IEEE Journal of Solid-State Circuits, vol.31, no.5, pp.677-688, 1996.

[37] A. Gerstlauer, S. Chakravarty, M. Kathuria, P. Razaghi. Abstract system-level models for early performance and power exploration, 17th Asia and South Pacific Design Automation Conference (ASP-DAC), pp.213-218, 2012.

Vita

Darshan Dhimantkumar Gandhi may be reached by the email address provided below.

Permanent email: darshang@utexas.edu

This thesis was typed by the author.