Temperature-Aware Test Scheduling for Multiprocessor Systems-On-Chip

David R. Bild*, Sanchit Misra*, Thidapat Chantem†, Prabhat Kumar*, Robert P. Dick*, X. Sharon Hu†, Li Shang‡, and Alok Choudhary*

*EECS Department
Northwestern University
Evanston, IL 60208, USA
d-bild@u.northwestern.edu
prabhat-kumar@northwestern.edu
{smi539, dickrp, choudhar}@eecs.northwestern.edu

†CSE Department
University of Notre Dame
Notre Dame, IN 46556, USA
{tchantem, shu}@nd.edu

‡ECE Department
University of Colorado at Boulder
Boulder, CO 80305, USA
li.shang@colorado.edu

***Abstract*—Increasing power densities due to process scaling, combined with high switching activity and poor cooling environments during testing, have the potential to result in high integrated circuit (IC) temperatures. This has the potential to damage ICs and cause good ICs to be discarded due to temperature-induced timing faults. We first study the power impact of scan chain testing for the ISCAS89 benchmarks. We find that the scan-chain test power consumption is 1.6× higher for at-speed testing than normal operating power consumption. We conclude that if the testing frequency is less than half of the normal frequency, then the testing power consumption may in fact be lower. However, due to differences in the cooling environments, the peak die temperatures may still be higher. Second, we present an optimal formulation for minimal-duration temperature-constrained test scheduling. Our results improve on the test schedule time of the best existing algorithm by 10.8% on average for a packaged IC thermal environment. We also present an efficient heuristic that generally produces the same results as the optimal algorithm, while requiring little CPU time, even for large problem instances.**

I. Introduction

The power densities of integrated circuits (ICs) have continually increased due to CMOS process scaling. Temperature depends strongly on power density [1]. Industry-accepted maximum operating temperatures exist for most types of packaged ICs [2]. Violation of these constraints during testing can lead to a number of problems. Elevated temperatures lead to thermal stress and accelerate failure processes such as electromigration, time dependent dielectric breakdown, and negative bias temperature instability, resulting in reduced reliability or permanent damage to ICs. Overheating can also lead to timing errors during the testing process, resulting in reduced yield.

Temperature is a concern during testing due to the potential for increased power consumption resulting from high switching activity [3]. Efficient testing requires exciting a chip with a large set of inputs in as short a time as possible. This is commonly accomplished through scan-chain testing [4]. The circuit is placed in a special test mode in which the flip-flops are serially connected in a fashion similar to a large shift register. Test patterns are shifted into the state elements serially. Thus, many state elements change value each cycle, and the combinational logic switching activity may increase relative to normal operation.

Ensuring that dangerous temperatures are not reached during test requires both an understanding of the test characteristics that lead to high temperatures and temperature-aware test scheduling methods. In this paper, we both characterize test power consumption and present a temperature-aware test scheduling algorithm for MPSoCs.

In the first part of this paper, we study the test and normal operation power consumptions for a set of benchmark circuits. In spite of the large quantity of power- and temperature-aware testing literature predicated on the intuition discussed above, there is very little published evidence for this supposed power consumption increase. In fact, dynamic power consumption is linearly dependent on both switching activity and clock frequency. Thus, although the switching activity may be higher during test, the power consumption may not be higher due to differences in the clock frequency. Due to limitations of the testing equipment, scan-chain testing is often performed at much slower speeds, e.g., 10–100 MHz, than is typical during normal operation, e.g., 500 MHz [5]. Consequently, the relative difference in power consumption between the test and normal operation modes is not clear. Therefore, the conditions under which testing may cause thermal problems are unknown.

Power- and temperature-aware test generation and scheduling may remain important even if the power consumption for typical-speed scan chain testing is not much greater than during normal operation. Many Built-In-Self-Tests (BIST) and timing-fault tests are performed at speed, using scan chain techniques. Although the power consumption may not be significantly higher during test, unsafe temperatures might be reached due to differences in the IC cooling environments. The cooling environment during normal operation is typically relatively good because several methods are used to decrease the thermal resistance between the die and the ambient environment [6]. The die is packaged with a thermally-conductive heat spreader and cooled by a heatsink. Typically, the thermal resistance between the heatsink and IC package is minimized by the application of a thermal paste. During test, however, the situation can be worse. For example, in pre-package test, thermal paste cannot always be used due to the difficulty in removing it [6]. Thus, elevated temperatures are still of concern.

In the second part of this paper, we develop and analyze a temperature-aware multiprocessor systems-on-chip (MPSoC) test scheduling algorithm. The goal is to minimize the amount of time required to perform testing while ensuring that temperature constraints are not violated. The MPSoC test scheduling problem is complicated by thermal interaction between processor cores. With the exception of resource conflicts, these cores can be tested concurrently in order to expedite the testing process. Resource conflicts can occur when the tests for two cores require access to the same physical resource (e.g., a bus or cache). This simultaneous testing leads to even higher power consumption than that observed during the test of single-core chips [7]. Parallel testing of two adjacent cores can lead to thermal hotspots because of the spatial concentration of the power dissipation.

II. Past Work and Contributions

In this section, we describe related past work and highlight our main contributions.

II.A. Power Analysis

Very little work comparing power consumption during test to that during normal operation exists in the published literature. Zorian an-

This work is supported in part by the SRC under award 2007-TJ-1589 and in part by the NSF under awards CCF-0702761, CNS-0347941, CNS-0410771, CNS-0404341, IIS-0536994, and CCF-0444405.

We would like to acknowledge Niraj K. Jha from Princeton University and Gokhan Memik from Northwestern University for their advice and support. We would also like to acknowledge Krishnendu Chakrabarty from Duke University and Bashir M. Al-Hashimi and Paul Rosinger from University of Southampton for their advice and access to their benchmarks.

alyzed the power consumption for a benchmark circuit, ASIC_Z [8]. He reports a normal operation power consumption of 0.9 W and an at-speed BIST test power consumption that is about $2.5\times$ higher: 2.3 W. This provides evidence that, if the test frequency is less than half the normal operating frequency, the power consumption might not be significantly higher during test.

Pouya and Crouch measured the power consumptions of compressed and uncompressed scan-chain test sets [9]. Although it was commonly believed that the compressed test set, with higher switching activity, would have much higher power consumption, this was not the case. They determined that the majority of the power consumption was due to the clock tree, specifically the clock inputs of the flip-flops. Because the clock inputs toggle at every clock transition, the power consumption was not affected by the differences in the combinational switching activity. The authors also state that tests with the ColdFire microprocessor core showed test power increases of $3\times$ over normal operation. They also mention that there have been reported cases where compressed test patterns have led to an $8\times$ increase in power consumption. However, they provide neither data nor references to support these claims.

Shi and Kapur report that historically, power consumption during scan-chain test is $3\times$ greater than during normal operation and that test patterns for some designs have lead to a $30\times$ increase [3]. However, they also do not provide data or references to support these claims.

Power Analysis Contributions: We provide a detailed study of the normal operation and scan-chain test power consumptions for the ISCAS89 benchmarks. We find that the switching activity is $4.1\times$ higher during test and that, for at-speed testing, the power consumption is only $1.6\times$ greater. This suggests that power consumption will be greater during scan-chain test only if the testing clock frequency is greater than approximately half the normal operating frequency.

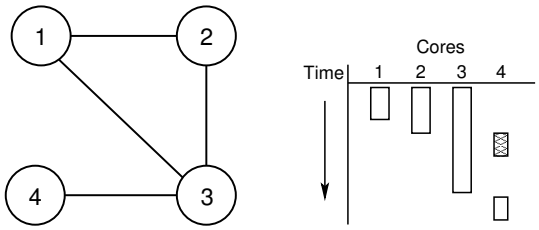
II.B. Test Scheduling

Several power-constrained approaches to test scheduling based on test reordering have been proposed in the literature. For single core systems, Flores et al. minimize the overall power consumption using test pattern sequence reorganization, taking into account don't-cares in the test vectors [10]. Girard et al. use test vector ordering to minimize the average power consumption [11]. Nourani and Chin introduce a Mixed Integer Linear Programming (MILP) based solution to perform power-time tradeoff analysis for SoC test schedules, allowing for a choice among several constraints including average and peak power consumption [12]. Iyengard and Chakrabarty present a technique for scheduling tests for SoCs under test precedence relationships and peak power constraints [13].

Minimizing peak power consumption does not necessarily minimize peak temperature because of spatial variations in the power and thermal profiles. High spatial power density concentrates heat generation, resulting in a high peak temperature and thermal hotspots.

Several papers describe methods to reduce peak temperature considering spatial and temporal power density variation. Liu, Veeraraghavan, and Iyengar present a method to reduce peak temperature and balance the thermal profile across a chip, assuming a limited-width test access mechanism [14]. They present a heuristic based on rectangle packing to minimize the testing time given a test access mechanism width and maximum temperature constraint. They present two algorithms that, starting from the heuristic test schedule, reduce the peak temperature and balance the thermal profile, at the expense of a 10.9% average increase in total schedule length. They do not provide an optimal formulation and thus cannot compare their final schedule lengths to the best possible under the reduced temperature constraint.

He et al. describe a combinatorial optimization-based method for reducing testing time, also assuming a limited-width test access mechanism, by partitioning and interleaving the test sequences for individual cores [15]. In cases where executing an entire test sequence for a core raises the peak temperature above the temperature bound, the sequence is split into portions small enough that the bound is not reached. The partitions for various cores are interleaved so that an individual core is allowed to cool between test partitions but the test access bus is still efficiently utilized. To avoid the need to integrate thermal analysis into their formulation, they assume that heat flow between neighboring cores is negligible and thus, for a given partition length, the temperature impact on a core can be precomputed.



(a) Test Compatability Graph (b) Test Schedule

Figure 1. Example schedule illustrating the weakness of the clique set scheduling method.

Rosinger, Al-Hashimi, and Chakrabarty present a technique to minimize testing time given thermal constraints and arbitrary resource conflicts [7]. They base their method on a minimal clique-set covering. The test set is represented as an undirected graph, with edges connecting tests that can be executed in parallel. A clique set of this graph thus consists of all possible combinations of test sequences that can be conducted concurrently. The execution time for a clique is taken to be the execution time of the longest test sequence in that clique. Their method finds the subset of cliques that covers all test sequences, never exceeds some temperature bound, and minimizes the total execution time.

Rosinger, Al-Hashimi, and Chakrabarty's technique requires that all test sequences in a chosen clique finish before another clique of test sequences can begin. Thus, if a clique consists of a short and long test sequence, test sequences that conflict with only the short test sequence cannot begin executing until both short and long test sequences have finished. Similarly, test sequences that are prevented by a temperature constraint only when both short and long test sequences are executing in parallel are forced to postpone execution until both have finished. This may lead to unnecessarily-long test times.

The total testing time can potentially be reduced if test sequences are allowed to begin as soon as this would not violate temperature and resource usage constraints, rather than waiting for the entire previous clique set to finish. This is illustrated in Figure 1. For cliques $\{1, 2, 3\}$ and $\{4\}$, the clique set based approach schedules test 4 only after test 3 completes. However, it could legally be scheduled after test 2 completes (the shaded box), which would lead to a shorter overall test sequence.

Test Scheduling Contributions: We propose a solution that allows tasks to begin at arbitrary start times. Our work makes the following main contributions:

1) An optimal formulation, including integrated thermal analysis, for test scheduling under thermal and resource constraints. This formulation permits arbitrary test start times and yields an average improvement of 10.8% and a maximum improvement of 36.7% in test time compared to past work.
2) A heuristic based on this formulation that handles large problem instances. The heuristic has an average increase in scheduling length of 0.5%, and at most 11.1%, relative to the optimal formulation. It has an average and maximum improvement of 10.5% and 36.7% over past work, and was never worse.

The rest of this paper is organized as follows. Section III compares test power consumption to normal operation power consumption. Section IV explains our temperature-aware MPSoC testing model. Section V gives an MILP formulation for the temperature-aware MPSoC test scheduling problem and compares its results with past work. Section VI presents and evaluates our proposed computationally-efficient heuristic.

III. Power Analysis

This section describes the normal operation and test mode power consumption analysis that we performed on the ISCAS89 benchmarks [16]. These benchmarks were chosen because they are representative of typical sequential logic designs, and thus are suitable for scan-based testing. Additionally, they have been widely used in computer-aided design research.

III.A. Experimental Setup

Each benchmark circuit was mapped to an industrial TSMC 65 nm standard cell library using Synopsys Design Compiler for a 2 ns clock period (500 MHz). To support scan testing, test_enable, test_input, and test_output pins were added. Using Synopsys DFT Compiler, all flip-flops were replaced with scan flip-flops and connected into a single scan-chain. For larger circuits, it is common for multiple scan-chains to be used to reduce the time necessary to load a test vector. However, that will not have a significant impact on the average power consumption; the majority of the state elements still toggle each cycle.

Test vectors were generated using Synopsys TetraMax ATG for a 10 ns test clock period (100 MHz) for the full fault list. Test pattern compression was not performed. A Verilog testbench suitable for cycle-accurate simulation of the test vector set was generated.

For normal operation, input vectors were generated randomly using a 10% switching probability for each primary input in the circuit. A random input vector served as the base and subsequent vectors were generated iteratively. For each circuit, the number of vectors generated (and thus the number of cycles simulated) was 500 times the number of primary inputs. To facilitate comparison with the test power numbers, the clock period was once again set to 10 ns.

Both the test and normal operation testbenches were simulated using Synopsys VCS, a cycle-accurate Verilog simulator. All value changes for each net were logged in the industry standard VCD format.

Power analysis was performed using Synopsys Primetime PX. Using power models supplied with the technology library and the switching traces recorded by the testbench simulation, the average power consumption for normal operation and scan test was determined for each circuit. The power consumptions, including leakage, internal, and switching, were calculated from cycle-accurate simulation traces.

III.B. Experimental Results and Analysis

The power analysis procedure was performed for 26 of the ISCAS89 benchmarks.[1] The average switching activities are also shown in Table I (Columns 5–7). As was expected, the switching activities are much higher during test, with an 8.5% average during normal operation and a 26.9% average during test. On average, switching activity increases by 4.1×. Table I also shows both the normal and scan power consumptions, as well as the increase (Columns 2–4). The average increase in power consumption is 1.6×.

Based on the average switching activities and the linear dependence of power consumption on switching activity, one might expect to see an approximate 4.1× increase in power consumption as well. The observed 1.6× increase is well under half of the expected increase. This discrepancy is due to the high power consumption of the clock tree [9]. The reported numbers represent the switching activity in the combinational logic. However, as mentioned earlier, a large fraction of the power consumption is actually due to the toggling of the flip-flop clock inputs. This power consumption is independent of the reported switching activity, and thus the 4.1× increase in switching activity results in only a 1.6× increase in power consumption.

The test and normal operation simulations, were performed at the same speed. The 1.6× increase implies that if the normal operating frequency is more than twice the testing frequency, power consumption will be higher during normal operation. We conclude that thermal problems will occur during scan-chain testing only in three circumstances.

1) The circuit is tested at a frequency greater than 1/2 normal operating frequency, e.g., during at-speed testing or built-in self test. This would result in higher power consumption than during normal operation.
2) The circuit has a greater inter-register combinational logic depth than the ISCAS89 benchmarks, resulting in a greater dependence of total power consumption on combinational switching activity. This would result in higher power consumption than during normal operation.
3) The circuit is tested in a thermal environment that is inferior to that used during normal operation, e.g., if the heatsink and fan used during testing results in a higher thermal resistance to the ambient than during normal operation. This would have little impact on power consumption, although it might indirectly increase leakage power. It would result in a higher temperature for the same power profile.

In the following sections, we present an optimal test scheduling formulation (Section V) and an efficient heuristic (Section VI), for use in minimizing test schedule length under a constraint on temperature when testing frequency, circuit structure, or testing thermal environment may produce thermal problems during test.

IV. System Model and Problem Definition

This section formally describes the test scheduling problem for MPSoCs with temperature and resource constraints.

IV.A. Test Model

Let $\mathcal{C}$ be the set of cores to be tested. For each core c in $\mathcal{C}$, $E(c)$ describes the execution time of c's test. $\Gamma(c_1, c_2)$ is a binary-valued function such that

$$\Gamma(c_1, c_2) = \begin{cases} 1 & \text{if cores } c_1 \text{ and } c_2 \text{ have a resource conflict} \\ 0 & \text{otherwise.} \end{cases}$$

Cores with resource conflicts may not be tested concurrently. T_{bound} is the maximum allowable temperature for any core during the testing process.

The objective is to determine the shortest test schedule that does not violate the temperature or resource constraints. More formally, we should find a test start time $t_s(c)$ for each $c \in \mathcal{C}$, such that no two cores c_1 and c_2 with $\Gamma(c_1, c_2) = 1$ are executing simultaneously, the chip peak temperature T_{max} is less than T_{bound}, and the latest finish time, $\max_{c \in \mathcal{C}}(t_s(c) + E(c))$, is minimized.

IV.B. Thermal Model

Rosinger, Al-Hashimi, and Chakrabarty's solution used external thermal modeling software, HotSpot [1], to compute the peak temperature at each stage of the optimization [7]. In contrast, we directly integrate the thermal model into an MILP-based formulation. The optimal solver thus has full knowledge of the temperature impact of all choices made during the optimization, potentially allowing for more efficient exploration of the solution space. MPSoC spatial power profiles vary over time. To capture this effect, we use a phased steady-state thermal model that evaluates the impact of each of the numerous

[1] Four of the thirty benchmark circuits did not compile successfully.

TABLE I
AVERAGE POWER CONSUMPTION AND SWITCHING ACTIVITY FOR ISCAS89 BENCHMARKS

Benchmark	Normal Power (W)	Scan Power (W)	Power Increase (×)	Normal Switching (%)	Scan Switching (%)	Switching Increase (×)
s9234	1.76e-04	3.20e-04	1.82	3.80	30.00	7.89
s38584	1.97e-03	3.13e-03	1.59	12.00	38.00	3.17
s838	3.60e-05	7.04e-05	1.96	2.90	24.00	8.28
s35932	2.70e-03	3.70e-03	1.37	17.00	39.00	2.29
s386	9.57e-06	1.13e-05	1.18	11.00	16.00	1.45
s526	3.00e-05	4.71e-05	1.57	10.00	32.00	3.20
s382	2.88e-05	4.54e-05	1.58	9.90	36.00	3.64
s1488	1.98e-05	3.03e-05	1.53	11.00	18.00	1.64
s1423	1.01e-04	1.72e-04	1.70	8.40	34.00	4.05
s953	3.78e-05	5.21e-05	1.38	5.90	12.00	2.03
s400	2.84e-05	4.53e-05	1.60	9.50	36.00	3.79
s5378	2.49e-04	3.70e-04	1.49	9.70	28.00	2.89
s641	2.50e-05	3.64e-05	1.46	8.20	17.00	2.07
s713	2.49e-05	3.66e-05	1.47	8.10	17.00	2.10
s349	2.14e-05	3.25e-05	1.52	11.00	31.00	2.82
s832	9.76e-06	1.64e-05	1.68	8.10	16.00	1.98
s298	2.22e-05	3.00e-05	1.35	16.00	32.00	2.00
s13207	7.78e-04	1.33e-03	1.71	5.60	35.00	6.25
s38417	1.95e-03	3.75e-03	1.92	2.50	43.00	17.20
s420	1.80e-05	3.38e-05	1.88	4.10	24.00	5.85
s15850	6.37e-04	1.16e-03	1.82	4.40	34.00	7.73
s510	6.84e-06	1.91e-05	2.79	3.30	21.00	6.36
s1238	3.10e-05	3.10e-05	1.00	9.00	7.10	0.79
s344	2.14e-05	3.29e-05	1.54	11.00	32.00	2.91
s820	9.99e-06	1.54e-05	1.54	8.00	14.00	1.75
s444	2.90e-05	4.33e-05	1.49	11.00	33.00	3.00

power profiles that occur in time, i.e., it considers the thermal effects of all concurrently-executing test sequences at each time instant. This phased steady-state formulation can be described by a linear system of equations.

Heat flow is modeled using the thermo-resistive model [17]. This is illustrated in Figure 2 for two cores. The arrows in the diagram map from physical regions in the IC to nodes in the model. For each core, the interface layer, heatspreader, and heatsink are lumped into one element. Additional neighboring cores are not shown, but would be modeled in a similar fashion. We model an MPSoC as a set of cores $\mathcal{C}$. For each core $c \in \mathcal{C}$, the length, width, height, and location are specified and the set of neighbors $\mathcal{N}_c$ is computed. Lateral thermal conductance between neighbors is represented by $G_N(c,n)$. The heatsink is modeled using discrete thermal elements corresponding to the locations of the underlying cores, with $G_H(c)$ representing the vertical thermal conductance between a core and its heatsink element. $G_{NH}(c,n)$ is the lateral conductance between adjacent elements. The interface layer is included in the heatsink elements because it is thin enough that lateral heat flow within it is negligible. Note that lateral heat flow in both the heatsink and silicon are modeled explicitly. $G_A(c)$ is the conductance to the ambient environment, which has temperature T_A. The power consumed by core c while running its associated test sequence is given by $P(c)$.

The thermal conductances are computed assuming a 0.6 mm thickness for the silicon die and 1 mm thickness for the copper heat spreader and heatsink. The conductivity of silicon is taken to be 148 W/mK and that of copper, 400 W/mK. The ambient temperature, T_A, is set to 45 °C.

Although the phased steady-state thermal model considers the effects of time-varying power profile on thermal profile, it does not model heat capacity. This is valid as long as the test sequence durations are much larger than the thermal RC time constant multiplied by the maximum power consumption variation, i.e., many hundreds of microseconds [18]. The test sequence lengths for our benchmarks range from 10 ms to 8,448 ms with an average time of 691 ms. Steady-state temperatures can only increase when the power consumption increases, which can happen only when a core begins to execute its test. Thus, for this phased steady-state analysis, the temperatures need only be computed at the start of each test. The temperature $T(c,t)$

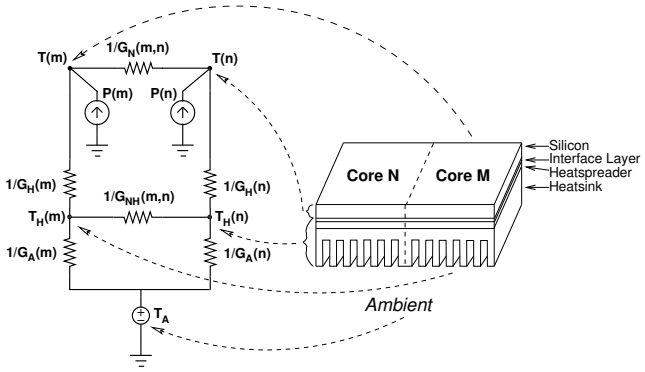


Figure 2. Thermoresistive model for two neighboring cores m and n. The arrows map from physical regions in the IC to the nodes in the thermoresistive model. For each core, the interface layer, heatspreader, and heatsink are lumped into one element.

of core c at time t can be computed as follows:

$$0 = G_H(c) \cdot [T(c,t) - T_H(c,t)] - \beta(c,t) \cdot P(c) + \sum_{n \in \mathcal{N}_c} G_N(c,n) \cdot [T(c,t) - T(n,t)] \quad (1)$$

$$0 = G_H(c) \cdot [T_H(c,t) - T(c,t)] + G_A(c) \cdot [T_H(c,t) - T_A] + \sum_{n \in \mathcal{N}_c} G_{NH}(c,n) \cdot [T_H(c,t) - T_H(n,t)] \quad (2)$$

where $T_H(c,t)$ is the temperature of the heatsink element corresponding to MPSoC core c at time t. For the phased steady-state model, the core temperature can only increase when a test sequence begins. Thus, $T(c_1, c_2)$ and $T_H(c_1, c_2)$ refer the core and heatsink temperatures of core c_1 at the time when c_2 begins its test sequence. $\beta(c,t)$ is 1 if core c is executing its test sequence at time t and is 0 otherwise.

In Section VI, we compare the temperatures calculated using this thermal model with those computed using publicly-available thermal modeling software. Those results show that this coarse-grained phased steady-state thermal model provides enough accuracy for the test scheduling problem.

TABLE II
TEST SCHEDULE LENGTHS

Design	Threshold Temp (°C)	Clique Set Test Length (s)	Optimal Test Length (s)	Improvement Over Clique Set (%)	Heuristic Test Length (s)	Increase Over Optimal (%)	Improvement Over Clique Set (%)
asic_z	55	0.323	0.204	36.7	0.209	2.5	35.3
asic_z	56	0.323	0.204	36.7	0.204	0.0	36.7
asic_z	57	0.282	0.191	32.3	0.191	0.0	32.3
asic_z	58	0.282	0.191	32.3	0.191	0.0	32.3
asic_z	59	0.282	0.191	32.3	0.191	0.0	32.3
kime	49	4.120	3.180	22.8	3.180	0.0	22.8
kime	50	3.490	3.180	8.9	3.180	0.0	8.9
kime	51	3.490	3.180	8.9	3.180	0.0	8.9
kime	52	3.490	3.180	8.9	3.180	0.0	8.9
kime	53	3.490	3.180	8.9	3.180	0.0	8.9
muresan_10	49	2.000	1.900	5.0	1.900	0.0	5.0
muresan_10	50	2.000	1.900	5.0	1.900	0.0	5.0
muresan_10	51	2.000	1.900	5.0	1.900	0.0	5.0
muresan_10	52	2.000	1.900	5.0	1.900	0.0	5.0
muresan_10	53	2.000	1.900	5.0	1.900	0.0	5.0
muresan_20	73	4.200	3.600	14.3	4.000	11.1	4.8
muresan_20	74	4.100	3.600	12.2	3.600	0.0	12.2
muresan_20	75	4.100	3.600	12.2	3.600	0.0	12.2
muresan_20	76	4.100	3.600	12.2	3.600	0.0	12.2
muresan_20	77	4.100	3.600	12.2	3.600	0.0	12.2
system_l	75	2.880	2.880	0.0	2.880	0.0	0.0
system_l	76	2.880	2.880	0.0	2.880	0.0	0.0
system_l	77	2.880	2.880	0.0	2.880	0.0	0.0
system_l	78	2.880	2.880	0.0	2.880	0.0	0.0
system_l	79	2.880	2.880	0.0	2.880	0.0	0.0
system_s	60	9.226	8.448	8.4	8.448	0.0	8.4
system_s	61	8.448	8.448	0.0	8.448	0.0	0.0
system_s	62	8.448	8.448	0.0	8.448	0.0	0.0
system_s	63	8.448	8.448	0.0	8.448	0.0	0.0
system_s	64	8.448	8.448	0.0	8.448	0.0	0.0

V. MILP FORMULATION

We have developed a mixed integer linear program to find an optimal solution to the test scheduling problem. This formulation draws from our previous work on real-time task scheduling [17]. The MILP formulation minimizes the total test schedule time while constraining the maximum temperature of any core and ensuring that no resource constraints are violated.

We define the following variables:

$$\eta(c_1, c_2) = \begin{cases} 1 & \text{if core } c_1 \text{ finishes testing before core } c_2 \text{ begins} \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(c_1, c_2) = \begin{cases} 1 & \text{if core } c_2 \text{ starts testing before } c_1 \text{ begins} \\ & \text{and overlaps the testing of core } c_1 \\ 0 & \text{otherwise} \end{cases}$$

The start and finish times for each core are represented by $t_s(c)$ and $t_f(c)$, respectively. The following constraint sets the finish time to the sum of the start time and the execution time.

$$\forall c \in \mathcal{C} : t_f(c) = t_s(c) + E(c)$$

As mentioned in Section IV-B, we need only sample the temperature at the start of each test. The linear temperature equations presented previously are included in the formulation without need of modification. Thus, the following relation ensures that the peak temperature T_{bound} is never violated.

$$\forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C} : T_{bound} \geq T(c_1, c_2)$$

The following constraints ensure that $\eta(c_1, c_2) = 1$ if and only if test c_1 finishes before test c_2 begins. In particular, the next constraint guarantees that for a pair of tasks at most one of the corresponding η variables is 1.

$$\forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C} : \eta(c_1, c_2) + \eta(c_2, c_1) \leq 1$$

The following constraint specifies that if the cores have a resource conflict, their execution times do not overlap. More specifically, if $\Gamma(c_1, c_2)$ is 1, the cores have a resource conflict and this constraint will ensure that one finishes its test before the other begins.

$$\forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C} : \eta(c_1, c_2) + \eta(c_2, c_1) \geq \Gamma(c_1, c_2)$$

The subsequent two constraints ensure that if $t_f(c_1) \geq t_s(c_2)$, then $\eta(c_1, c_2)$ is set to 0 and vice versa. λ is a constant larger than the longest possible test sequence. In our case, λ can be set to the sum of all the test execution times.

$$\forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C} : t_f(c_1) \leq t_s(c_2) + (1 - \eta(c_1, c_2)) \cdot \lambda$$

$$\forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C} : t_s(c_2) \leq t_f(c_1) + \eta(c_1, c_2) \cdot \lambda$$

The next set of constraints ensure that $\beta(c_1, c_2) = 1$ if and only if $t_s(c_2) \leq t_s(c_1) \leq t_f(c_2)$. Specifically, the next two constraints force $\beta(c_1, c_2)$ to be 0 if either of the two inequalities does not hold.

$$\forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C} : t_s(c_1) \leq t_f(c_2) + (1 - \beta(c_1, c_2)) \cdot \lambda$$

$$\forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C} : t_s(c_2) \leq t_s(c_1) + (1 - \beta(c_1, c_2)) \cdot \lambda$$

The following constraint forces $\beta(c_1, c_2)$ to 1 if both do hold. ϵ is small constant used to ensure that floating-point calculation errors do not cause contiguous tasks to appear to overlap in time.

$$\forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C} : t_s(c_1) - \eta(c_2, c_1) \cdot \lambda \leq t_s(c_2) + \beta(c_1, c_2) \cdot \lambda - \epsilon$$

For the sake of consistency, we let

$$\beta(c_1, c_1) = 1$$

We minimize the total test schedule length, or equivalently, the latest test finish time by introducing a variable t_{finish}, which represents the total time for the test schedule.

$$\forall c \in \mathcal{C} : \; t_{finish} \geq t_f(c)$$

$$\text{minimize } t_{finish}$$

Optimal Results

The optimal formulation was solved using AMPL and CPLEX for the same set of MPSoC designs used by Rosinger, Al-Hashimi, and Chakrabarty for their clique set technique [19], [20]. For each benchmark, the set of resources or cores, the floorplan, the test sequence lengths, the test sequence power consumptions, and resource conflicts are specified. The thermal conductivities for the thermal model presented in Section IV-B are computed based on the floorplan. Each design was evaluated for five different temperature bounds in 1°C increments. Multiple evaluations are used to provide a better sense of the discontinuous relationship between temperature bounds and optimal schedule lengths. For each design, bounds were selected that highlights this relationship.

In order to compare our results with prior work on the clique set formulation [7], we reimplemented the technique using an MILP formulation with our thermal model. For a given design, all cliques of the resource constraint graph are precomputed. The phased-steady state thermal model is implemented in the formulation to calculate the peak temperature for each clique. From the subset of cliques that do not violate the thermal constraint, the formulation chooses those that fully cover the set of tests and minimizes the total test schedule length.

The results are shown in Table II. Our optimal formulation yields average and maximum improvements in test schedule length of 10.8% and 36.7% compared to the clique set technique. For most designs and temperatures evaluated, our MILP formulation yields a test schedule limited only by the resource conflicts (i.e., increasing the temperature bounds further will not improve the test schedule). In the case of design system_l, no improvement is seen for any temperature. This design is severely resource-constrained; the majority of the cores are compatible with only one, if any, of the other cores. For such a design, the clique sets will contain only a single test, and thus the two approaches will yield identical results.

Extensions to the Model

Our choice of benchmarks in the preceeding section was motivated by our comparison with the prior work. More realistic designs would have a finer granularity for the test sequences and would include the number of physical resources on the Automated Test Equipment (ATE) necessary for each test. For example, the number of probe pins available is a physical limitation of the test equipment, and so provides an additional constraint on the sets of tests that can be scheduled concurrently. However, the proposed technique is applicable to these more realistic circuits with few or no modifications.

Although for the benchmarks we used in our experiments each resource had a single atomic test sequence, some real test sequences are composed of several shorter sequences that need not be executed consecutively. Our technique can easily handle this with no modification. The test sequence for a given core is first split into its shortest (atomic) subsequences. Resource conflicts are added between these subsequences and each subsequence is scheduled as a separate test.

The probe pin limitation can be easily handled with the addition of a single constraint. We assume that the test access mechanism for each core or resource is predetermined and thus the number of probe pins required for each test sequence is already known. Given the number of pins available on the ATE, Ψ_{max}, and the number of pins required by each test, $\Psi(c)$, the following constraint ensures the number of pins in use is never greater than the total number of pins.

$$\forall c_1 \in \mathcal{C} : \Psi_{max} \geq \sum_{c_2 \in \mathcal{C}} \beta(c_1, c_2)\Psi(c_2)$$

VI. Heuristic

The MILP formulation of the problem is guaranteed to find an optimal solution, but the problem is $\mathcal{NP}$-hard by reduction from task scheduling. Therefore, this formulation can only provide solutions to small problem instances. For design muresan_20, with 20 cores, the solver took about 30 minutes. The computation time for solving this optimal formulation for future many-core MPSoCs may be prohibitive. Thus, a computationally-efficient heuristic is needed.

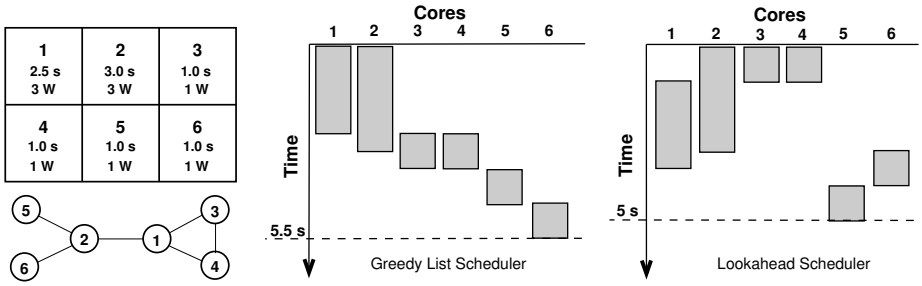
Algorithm 1 TestSched()

```
1: cur_time ← 0
2: running_cores ← ∅
3: verify that each core can be scheduled in isolation
4: if any core cannot be scheduled in isolation then
5:    halt
6: end if
7: while there are unscheduled cores do
8:    seed_cores ← all unscheduled cores which could be legally
      scheduled
9:    to_sched ← ∅
10:   if seed_cores ≠ ∅ then
11:      temper_order ← unscheduled cores ordered by increasing
         temperature impact
12:      for each seed ∈ seedCores do
13:         seed_groups[seed] ← seed
14:         for each c ∈ temperOrder do
15:            if c can be legally scheduled then
16:               seed_groups[seed] ← seed_group[seed] ∪ c
17:            end if
18:         end for
19:      end for
20:      sorted ← seed_groups sorted by decreasing size, then temp
21:      to_sched ← first seed in sorted
22:   end if
23:   if to_sched = ∅ then
24:      update cut_time to when next test finishes
25:   else
26:      schedule to_sched, adding it to running_cores
27:   end if
28: end while
29: return SCHEDULE
```

For their clique set problem formulation, Rosinger, Al-Hashimi, and Chakrabarty proposed a heuristic that iteratively builds cliques of concurrent tests. These cliques or test sets are built by adding tests one-by-one while ensuring that resource constraints are not violated. Because they use an external thermal modeling tool, HotSpot, verifying that the temperature constraint is not violated after the addition of each test is prohibitively expensive. To overcome this problem, they developed a thermal cost model that attempts to predict the thermal impact of adding a test to a set. However, because the thermal cost model is not precise, it can overestimate the peak temperatures resulting in longer test schedules.

Because we are using a phased steady-state thermal model, which is quickly solvable using basic linear algebra, we are able to integrate thermal analysis into our heuristic. This integration avoids both slow external calls to thermal modeling software and the need for a thermal cost model, allowing for rapid thermal analysis. Consequently, our heuristic has full knowledge of the temperature impacts of all choices made during the scheduling process. This knowledge allows for precise and more efficient exploration of the search space resulting in shorter test schedules.

At first glance, it may appear that this scheduling problem is easily solved by common scheduling heuristics. However, due to the complex interaction of the thermal constraints and resource conflicts, simple scheduling heuristics do not perform well for this problem. We first describe a greedy list scheduler-based approach and explain properties of the problem that prevent near-optimal test schedules. We then develop our improved heuristic based on these observations.



(a) Floorplan and Test Compatability Graph

(b) Schedules

Figure 3. Example schedule illustrating the benefit of the lookahead heuristic.

List Scheduler: Based on the intuition that tests with greater temperature impact (i.e., increase in peak temperature) should be scheduled early to maximize the chance that a temperature-compatible test will still be available to schedule concurrently, it seems that a greedy list scheduler, with the tests sorted in decreasing order of temperature impact, may be sufficient. However, this method does not perform well because it neglects future effects on resource conflicts due to scheduling a test sequence for a certain core.

An example of this is shown in Figure 3. The example MPSoC consists of six cores, each with a single test. The floorplan, test compatability graph, and test power consumptions and lengths are shown in Figure 3(a). Cores 1 and 2 have much higher power consumptions than the other cores, and thus will have higher temperature impacts. Consequently, they are scheduled first by the list scheduler, as shown in Figure 3(b). Unfortunately, as mentioned above, this decision is made without considering the resource conflicts. In this design, cores 2, 3, and 4 could be scheduled concurrently. However, scheduling cores 1 and 2 at the same time prevents cores 3 and 4 from performing their entire test sequences in parallel with core 2.

Lookahead Heuristic: We propose a heuristic incorporating a lookahead scheme to make a selection. Because the lookahead heuristic takes into account the impacts on both temperature and resource conflicts, this approach allows for a good trade-off between computational complexity and the quality of the generated test schedule. This improvement is illustrated for the prior example in Figure 3. The lookahead heuristic identifies the future resource conflicts caused by scheduling cores 1 and 2 together, and instead first schedules cores 2, 3, and 4. This allows for full concurrent execution of these test sequences, leading to an improved (and in this case, optimal) test length.

For a given problem instance, the algorithm first checks that each core can be scheduled in isolation without violating the temperature constraint (line 3). If a test sequence cannot be scheduled alone, then there is no legal schedule and the designer must raise the temperature bound, reduce the power consumption of the violating cores, or improve the cooling solution. If a legal schedule is possible, the algorithm proceeds.

The algorithm iteratively schedules tests in the following manner. During each iteration, the remaining unscheduled tests are checked to determine which can be started at the current time without violating any resource constraints with the already executing cores or violating the temperature constraint (line 8). If no tests can be scheduled at the current time (line 10), the time is updated to the next time that a currently running test finishes (line 24). Each test that can be scheduled is used as a *seed* to generate a group of unscheduled tests that could be scheduled together. These groups are used to provide a measure of the future impact of scheduling the associated seed test at the current time. The seed with the best group is scheduled and the process is repeated. Note that only this seed, and not its entire group, is scheduled.

To build a group, the unscheduled tests are ordered by increasing peak temperature, if that test and the seed were scheduled (line 11). Tests are then added to the group in order from the sorted list (line 14). A test that violates a resource constraint cannot be added and any test that violates the temperature constraint cannot be added (line 15). In this way, each group is built to approximate the maximum-size set of tests that can be started at the current time, given that the seed test is scheduled. The seed with the largest group is then scheduled (lines 20-21). Ties in group size are broken using the temperature impact of the entire group. If a tie still remains, the seed for the group, among the largest groups, with the highest peak temperature is chosen and scheduled.

The preceding algorithm computes peak temperatures by expressing Equations 1 and 2 of the phased steady-state thermal model as a system of equations of the form $\mathbf{A} \times T + B = 0$, with size $(2 \times |\mathcal{C}|)^2$, where $\mathbf{A}$ is a matrix of thermal conductances and B is a vector of power consumptions for operating cores. Because the matrix $\mathbf{A}$ is fixed by the floorplan, its inverse need only be computed once. Only the power vector B needs to be updated for each calculation. The temperature vector T can thus be computed with a matrix multiplication. Consequently, the time complexity of the heuristic is $\mathcal{O}(|\mathcal{C}|^3 \cdot |\mathcal{C}|^2)$ or $\mathcal{O}(|\mathcal{C}|^5)$.

Heuristic Results

The heuristic was implemented in Python and tested on the same designs as the optimal MILP formulation, using the packaged thermal model parameters. Evaluating these 30 problem instances (6 designs at 5 temperature bounds each) took 10.5 s, or on average 0.3 s each, on a 2.1 GHz AMD Athlon XP 3000 processor with 2.5 GB of RAM. Based on these running times and the algorithm's complexity, it should scale for MPSoCs with hundreds of cores.

A comparison of the schedule lengths generated by the two methods is shown in Table II. For these data, the heuristic produces schedules with finish times within 0.5% of the optimal schedule on average, and within 11.1% of the optimal solution in the worst case. For four of the six designs, the heuristic returns optimal schedules for all five temperature bounds tested. It never returns a schedule length that is longer than the clique set method and yields an average improvement of 10.5% compared to the best existing prior work.

Dynamic Thermal Effects

It is possible for the steady-state peak temperature to be slightly less than the dynamic or transient peak temperature, due to *dynamic thermal effects*. The temperatures of real materials change gradually, even in response to an instantaneous change in power consumption. Consider two adjacent cores M and N. At time t, M finishes its test sequence and N begins its sequence. Although both cores are never executing tests at the same time, just after time t, core N will be heating up and core M will still be hot. Thus, as the system transitions from the steady-state temperatures for the test of core

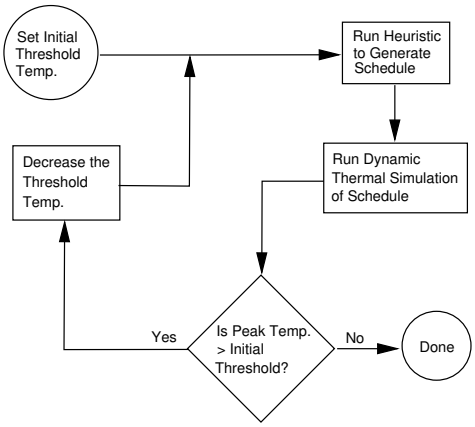


Figure 4. Algorithm to ensure test schedules never violate the temperature constraint due to dynamic effects.

M to the steady-state temperatures for the test of core N, the peak temperature could temporarily increase above either of the steady-state peak temperatures.

This potential increase in peak temperature will generally be quite small. To confirm this, we performed dynamic thermal analysis on the test schedules generated by the heuristic using publicly-available thermal modeling software [21]. As expected, the peak temperature when considering dynamic effects never exceeded the phased steady-state peak temperatures by more than 0.5 °C. Although this indicates that it should be safe to ignore dynamic thermal effects during test scheduling, for completeness we provide a modification to the heuristic that ensures that the dynamic peak temperatures will not violate the temperature constraint.

The proposed algorithm is illustrated in Figure 4. The method uses the phased steady-state-based heuristic to quickly generate the test schedule. After a schedule has been generated, it is checked, using a dynamic thermal analysis tool, to ensure that it does not violate the initial temperature bound due to dynamic effects. If it honors the bound, the algorithm terminates and returns a schedule. If the temperature does violate the initial constraint, the temperature bound is reduced by some small value (e.g., 0.1 °C). The heuristic is run again and the new schedule is checked against the *initial* temperature bound. This process is repeated until a satisfactory schedule has been found. Because the steady-state and dynamic peak temperatures will be quite similar, only a few iterations should be needed to find a schedule that meets the constraint.

VII. Conclusions

This paper first presented a comparison of power consumption between scan-chain test and normal operation. Although the switching activity was 4.1× higher during test, the power consumption was only 1.6× higher at the same frequency. Based on these data, we indicated the circumstances in which temperatures during test may exceed those during normal operation.

In order to handle testing scenarios in which temperature is a concern, we developed an optimal MILP formulation for the MPSoC test scheduling problem. This formulation minimizes test schedule time under a peak temperature constraint in the presence of resource conflicts. Given the same temperature constraint, it improves schedule lengths given by the best existing approach by an average of 10.8%. We have also presented a computationally-efficient heuristic that produces test schedules that honor temperature bounds and are within 0.5% of the optimal schedule durations on average. Both our MILP formulation and heuristic consider time-varying power profiles and spatial thermal variation.

When choosing the next test to schedule, the heuristic must estimate the future impacts on temperature and resource conflicts of scheduling each test. Therefore, our heuristic employs a seed-based approach. Each test is used as a seed to grow a cluster of tests that might be scheduled concurrently. The seed for the best cluster is then scheduled. In all cases, the heuristic outperforms the best existing work.

References

[1] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. Int. Symp. Computer Architecture*, June 2003, pp. 2–13.

[2] "Understanding integrated circuit package power capabilities," National Semiconductor, Tech. Rep., Apr. 2000, http://www.national.com/ms/UN/UNDERSTANDING_INTERGRATED_CIRCUIT_PACKAGE_POWER_CA.pdf.

[3] C. Shi and R. Kapur, "How power-aware test improves reliability and yield," Sept. 2004, http://www.eetimes.com/showArticle.jhtml?articleID=47208594.

[4] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. IEEE Press, 1990.

[5] S. Chakravarty, LSI Corp., personal communication, Feb. 2008.

[6] P. Tadayon, "Thermal challenges during microprocessor testing," *Intel Technology Journal*, vol. 4, no. 3, Aug. 2000.

[7] P. Rosinger, B. Al-Hashimi, and K. Chakrabarty, "Thermal-safe test scheduling for core-based system-on-chip integrated circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 2502–2512, Nov. 2006.

[8] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices," *IEEE VLSI Test Symposium*, pp. 4–9, Apr. 1993.

[9] B. Pouya and A. L. Crouch, "Optimization trade-offs for vector volume and test power," in *Proc. Int. Test Conf.*, Oct. 2000, pp. 873–881.

[10] P. Flores, J. Costa, H. Neto, J. Monteiro, and J. Marques-Silva, "Assignment and reordering of incompletely specified pattern sequences targetting minimum power dissipation," in *Proc. Int. Conf. VLSI Design*, Jan. 1999, pp. 37–41.

[11] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac, "Reduction of power consumption during test application by test vector ordering," *Electronics Ltrs.*, pp. 1752–1754, Oct. 1997.

[12] M. Nourani and J. Chin, "Test scheduling with power-time tradeoff and hot-spot avoidance using MILP," *Computers and Digital Techniques, IEE Proceedings*, vol. 151, no. 5, pp. 341–355, Sept. 2004.

[13] V. Iyengar and K. Chakrabarty, "System-on-a-chip test scheduling with precedence relationships, preemption, and power constraints," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 9, pp. 1088–1094, Sept. 2002.

[14] C. Liu, K. Veeraraghavant, and V. Iyengar, "Thermal-aware test scheduling and hot spot temperature minimization for core-based systems," in *Proc. Int. Conf. Defect and Fault Tolerance in VLSI Systems*, Oct. 2005, pp. 552–562.

[15] Z. He, Z. Peng, P. Eles, P. Rosinger, and B. M. Al-Hashimi, "Thermal-aware SoC test scheduling with test set partitioning and interleaving," in *Proc. Int. Conf. Defect and Fault Tolerance in VLSI Systems*, Oct. 2006, pp. 477–485.

[16] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Prof. Int. Symp. Circuits and Systems*, May 1989, pp. 1929–1934.

[17] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," in *Proc. Design, Automation & Test in Europe Conf.*, Mar. 2008.

[18] M. Barcella, W. Huang, K. Skadron, and M. Stan, "Architecture-level compact thermal R–C modeling," University of Virginia Dept. of Computer Science, Tech. Rep., July 2002.

[19] "ILOG CPLEX," ILOG, Inc., http://www.ilog.com/products/cplex.

[20] P. Rosinger, "Sample designs for validating thermal-aware test solutions," 2005, http://users.ecs.soton.ac.uk/pmr/thermal_test_sample_designs.zip.

[21] Y. Yang, Z. P. Gu, C. Zhu, R. P. Dick, and L. Shang, "ISAC: Integrated space and time adaptive chip-package thermal analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Jan. 2007.