Fighting Dark Silicon: Toward Realizing Efficient Thermal-Aware 3-D Stacked Multiprocessors

Sumeet S. Kumar, *Member, IEEE*, Amir Zjajo, *Member, IEEE*, and Rene van Leuken, *Member, IEEE*

***Abstract*— This paper investigates the challenges of dark silicon that impede the performance and reliability of 3-D stacked multiprocessors. It presents a multipronged approach toward addressing the thermal issues arising from high-density integration in die stacks, spanning architectural techniques, design methodologies, and runtime temperature management. Importantly, this paper provides novel insights into the causes of hotspot formation in 3-D ICs and details a practical approach toward exploring and mitigating performance-limiting thermal behavior early in the system design flow.**

***Index Terms*— 3-D integrated circuits, design for quality, memory management, multicore processing, system-level design.**

I. Introduction

3-D INTEGRATION provides a promising path toward realizing densely integrated multiprocessor systems, in the form of a stack of silicon dies. The dies are interconnected by means of vertical metal wires known as *through silicon vias (TSVs)*. Although 3-D integration allows compute power and system heterogeneity to be scaled up to meet the growing needs of applications, die stacks exhibit complex thermal behavior that can be detrimental to both performance and reliability. Power dissipation by components in ICs results in the generation of heat, which is typically conducted to the ambient by a heatsink. Nagata [1] determined that the maximum allowable power dissipation in an IC is limited by its physical construction and the thermal efficiency of its cooling interfaces. This limitation best describes the phenomenon termed as *dark silicon*, a reference to the large sections of modern ICs powered down to prevent overheating. The complex thermal behavior of 3-D ICs further compounds this issue, with each silicon die adding up to 12 layers of varying composition and thickness to the heat flow path from power dissipating elements. This gives rise to a number of issues that adversely impact on performance and accelerate the degradation of devices [2]. First, since operating constraints within the die stack vary due to the dependence of thermal conductance on the length of the heat flow path, temperature margins are not uniform throughout the stack. Each *processing element (PE)* in the system thus has a different operating envelope that is determined by its physical location. Second, the complex interdependence of execution performance, power, and temperature, together with the consequences of thermal coupling between otherwise independent elements of the system, yields an extremely nontrivial design space. This makes achieving optimal system performance and reliability difficult. Third, the stacking of power dissipating elements results in increased power density, causing elevated operating temperatures and steep thermal gradients. In addition to these challenges, power dissipation is in itself a critical factor in thermally constrained systems. Consequently, an increase in effective performance through scaleup of system size can only be achieved by concomitantly improving energy efficiency.

Prior art in addressing the issue of dark silicon typically focuses on a limited subset of these outlined challenges. In their seminal work, Skadron *et al.* [3] established the fundamentals of thermal-aware microarchitecture design, proposing temperature management strategies for planar 2-D ICs and the *Hotspot* thermal simulator. Puttaswamy and Loh [4] subsequently investigated the 3-D design of a single processor core, mitigating dark silicon through an approach that balanced power density through optimizations and design partitioning of the pipeline. Coskun *et al.* [5] evaluated the global implications of the 3-D design space by analyzing the cost-temperature tradeoffs for floorplanning strategies in the context of a multiprocessor. In [6], we explored the impact of TSV placement topologies on the electrical performance of vertical interconnect and its area overheads. Similarly, a number of other works proposed techniques for thermal-aware 3-D floorplanning [7], [8] and thermal-aware path finding [9], [10]. Runtime temperature management was notably studied by Sridhar *et al.* [11], in the context of a liquid cooled multiprocessor. Additionally, their efforts yielded the 3-D-ICE thermal simulator, which is used in an adapted form in this paper. From the architecture perspective, in their evaluation of the primary sources of inefficiency in chip multiprocessors, Hameed *et al.* [12] demonstrated that the largest impediment to efficient scaling lies in architectural overheads, dominated by the memory hierarchy. This is evident not only in the energy breakdown that they provide for their multiprocessor system but also in the case of recently fabricated prototypes such as the thousand-processor KiloCore array [13].

In this paper, we detail a comprehensive design trajectory that holistically addresses the previously outlined challenges of dark silicon and examines how these impede the performance and reliability of 3-D stacked multiprocessors. We describe a practical approach toward overcoming thermal issues arising from high-density integration, spanning the complete design space ranging from architectural techniques and system design methodologies to runtime resource management. This results

Manuscript received July 8, 2016; revised November 7, 2016; accepted December 9, 2016. Date of publication January 24, 2017; date of current version March 20, 2017. This work was supported by the CATRENE programme's Computing Fabric for High Performance Computing under Project CA104.

The authors are with the Circuits and Systems Group, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: s.s.kumar@tudelft.nl; a.zjajo@tudelft.nl; t.g.r.m.vanleuken@tudelft.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2016.2642587

in the maximization of system performance and a concomitant minimization of dark silicon. This paper provides a unique top-to-bottom view of the dark silicon problem and embodies a cohesive approach toward mitigating its effects in 3-D ICs. This paper departs significantly from the silo-based view adopted by previously published literature and is novel in its holistic approach to the challenge.

II. Overview

The challenges of dark silicon are best illustrated by Nagata's equation [1], which describes the relationship between power dissipation in ICs and the efficiency of their thermal interface with the ambience. This is given by

$$\frac{\alpha N_G E}{t_{\text{pd}}} \leq g \cdot \Delta T \quad \text{where } g = \kappa \frac{A}{l}. \tag{1}$$

This relation indicates that the number of gates (N_G) with activity rate (α), energy dissipation (E), and clock period (t_{pd}), which can be integrated within a single IC, is limited by the thermal conductance (g) of its interface with the ambience as well as the ambient temperature. ΔT is the maximum permissible difference between on-chip and ambient temperatures and represents the ideal temperature margin available at zero power dissipation. The actual available margin varies with distance from the heatsinking surface, as observed in the expansion of g in terms of thermal conductivity (κ), surface area (A), and distance from the site of power dissipation (l). For this reason, deep tiers in 3-D ICs exhibit lower utilization and hence lower performance. The relation (1) provides a concise description of the fundamental challenge that is dark silicon and the parameters that contribute to its manifestation. The values of these parameters are a consequence of choices made at various stages of the design process, and thus form control knobs that can be used to influence the system's thermal behavior. The knobs are categorized according to the design stage at which they can be exercised into the following.

1) *Architecture:* Energy minimization (E), energy density reduction (E/A), and latency minimization ($\Sigma t_{\text{pd}}/operation$).
2) *System Design:* Temperature-margin-aware power balancing (E, $\kappa A \Delta T/l$).
3) *Runtime:* Activity balancing ($\alpha N_G E/A$) and dynamic power performance tuning ($\Delta E/t_{\text{pd}}$).

Consequently, the solution to the dark silicon problem is threefold. First, realizing large-scale heterogeneous systems within the tight area constraints of die stacks requires drastic improvements to efficiency, i.e., effective execution performance per watt. This necessitates the use of architectural techniques to reduce the energy footprint of operations. Second, power density and nonuniform temperature margins are the key factors that determine operating temperatures within die stacks and thus limit the effective utilization of components. Optimization of performance is predicated upon the accurate modeling of thermal behavior and the early detection and mitigation of thermal issues. Third, runtime variations in execution characteristics make it essential to continually monitor and adapt the power dissipation of components. For a 3-D integrated system to be reliable, it is important that runtime power management and activity control mechanisms continually maintain operating temperatures and gradients within safe margins. Each of these aspects is examined individually in Sections III–V.

III. Architectural Efficiency

Naga is a large-scale heterogeneous 3-D stacked multiprocessor platform that we built for the acceleration of multimedia and general computing workloads. It consists of a shared memory cluster (*NagaS*) and a message-passing dataflow cluster (*NagaM*), both managed by an array supervisor and interconnected over the best effort 3-D *network-on-chip (NoC)* [14]. The memory subsystem consists of a two-level memory hierarchy with a nonuniform cache architecture for second-level cache banks in NagaS and distributed memories backed by high-throughput streaming data buffers for NagaM. Naga's heterogeneous architecture encapsulates the wide design space of modern multiprocessors and provides a generalized broadly applicable test vehicle for the concepts proposed in the rest of this paper. Its architecture is illustrated in Fig. 1(a).

The aggregate runtime of a task constitutes the execution time of its instructions and the time spent in performing memory load stores (t_{memory}). In multiprocessors, it is the latter that acts as a critical bottleneck for scalability, growing in magnitude with system size. Its impact is largely influenced by the efficiency of on-chip data storage and transfer, which in the case of cache-based multiprocessors is reflected in *average memory access time (t_{AMA})*

$$t_{\text{memory}} = M_A \cdot t_{\text{AMA}} \tag{2}$$

where

$$t_{\text{AMA}} = \mu_{\text{hit}} \cdot t_{\text{hit}} + \mu_{\text{miss}} \cdot t_{\text{miss}}. \tag{3}$$

M_A refers to the total number of memory accesses made by the executing application and μ_{hit} and μ_{miss} are the hit and miss rates of the data cache with the latencies t_{hit} and t_{miss}, respectively. t_{AMA} encapsulates the cost of accessing data from the local cache (t_{hit}) and the cost of fetching data from a remote cache in the event of it not being available locally (t_{miss}). Improving execution performance for cache-based multiprocessors necessitates efficient handling of on-chip data so as to minimize t_{AMA} by reducing t_{hit}, t_{miss}, and μ_{miss}. Furthermore, improving energy per access (E_{AMA}) requires a corresponding reduction in the energy consumed per hit and per miss.

In message-passing architectures, communication occurs in the form of messages between concurrently executing tasks. Message-passing architectures [15]–[17] typically require communications to be explicitly managed through the use of special functions in their software libraries. Thus, t_{memory} for such multiprocessors consists of not only the time incurred in performing the actual data transfer (t_{transfer}) but also the time spent in executing message-passing library functions (t_{mp})

$$t_{\text{memory}} = t_{\text{mp}} + t_{\text{transfer}} + t_{\text{fc}}. \tag{4}$$

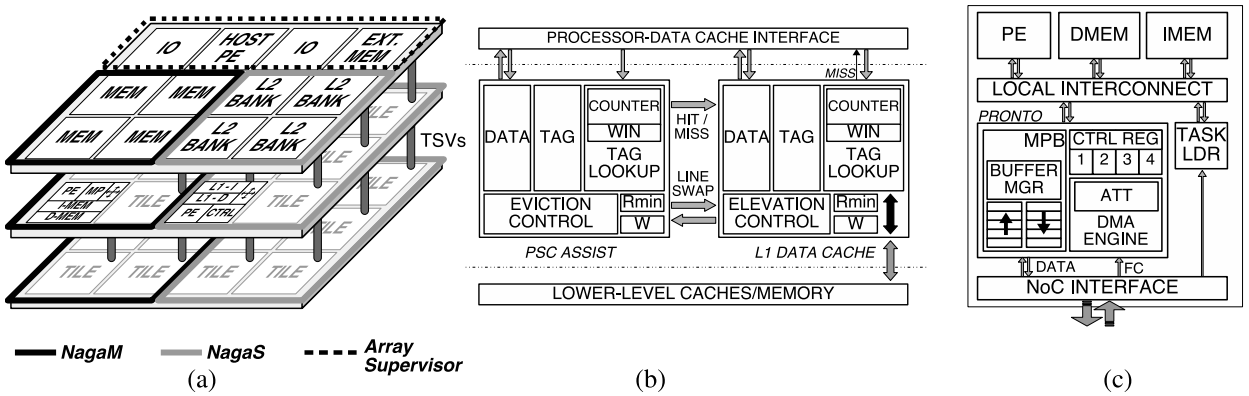

Fig. 1. (a) Illustration of the *Naga* multiprocessor with the shared memory *NagaS* and message-passing *NagaM* subarrays. (b) Architecture of the PSC assist cache and its interface for cache line swaps with L1D. (c) NagaM tile containing a PE, local memories, a Pronto message-passing interface, and a network interface.

An additional overhead t_{fc} is incurred in synchronizing and implementing flow control between the communicating tasks. The magnitudes of t_{mp} and t_{fc} together represent the overheads for message transfers and determine the efficiency of the message-passing implementation.

A. Minimizing Latency and Energy in Data Caches

Assist caches are small memory buffers that augment the capacity of the *first-level data cache (L1D)* in order to improve performance and energy consumption. Their small size affords them excellent access latencies with extremely small energy overheads. However, in order to profitably leverage these characteristics, it is essential to maximize the number of memory references that incur the short access costs of the assist, while simultaneously avoiding any adverse impact on the L1D. This is where existing proposals fail, as they either present benefits only for a minor fraction of all memory references, thereby diminishing their effective impact, or induce side effects in the system that nullify their advantage. The state-of-the-art *victim cache (VC)* [18] and *selective VC (SVC)* [19] belong to the former category, while *HitME* [20] and Duong's *L0* scheme (*I1P101* in particular) [21] belong to the latter. The VC, for instance, reduces t_{AMA} by minimizing the cost of L1D conflict misses. However, since it does not impact t_{hit}, its effective performance benefit is small and restricted only to cache conflicts. Conversely, in the case of the L0, although t_{hit} is minimized for every memory reference, the necessity of moving cache lines to and from the assist results in drastically higher energy per access and aggravated cache contention. We conclude that in order to maximize the performance benefits of such schemes, it is essential to implement some form of selectivity in what data are cached in the assist and to ensure that these data are sufficiently reusable so as to minimize t_{AMA} and energy per access E_{AMA}.

Access persistence is defined as the number of accesses (R) made to a line within a certain fixed window of data memory references (W) by the processor. The persistence of a line is indicative of its reusability, since a line that is frequently accessed in a short span of time is likely to be referenced again in the near future. We propose *persistence selective caching (PSC)*, an assist caching scheme that uses access persistence as the criterion for admitting cache lines into the assist. Fig. 1(b) illustrates a PSC-assisted L1D cache. Both the assist and L1D are probed in parallel on references by the processor. Since the two are mutually exclusive in their content, cache lines may reside in either cache but not both simultaneously. Consequently, a reference to a line may hit in either cache or miss in both, with the latter case serviced through the L1D alone. Following each cache hit, persistence data for lines are updated in the tag array and lines with sufficient persistence are moved (elevated) to the assist cache. The threshold level of access persistence that lines require is specified as the minimum number of hits (R_{min}) that a line must have in a contiguous sequence of W memory references. The W parameter lends reuse measurements a characteristic impermanence, ensuring that only the most active data sets within each phase of execution are tracked. Sequences are tracked by time stamping lines during updates to their persistence data. This allows stale reuse data to be efficiently invalidated without having to frequently refresh the entire cache. The persistence threshold is configurable and is set using control registers within the L1D and assist interfaces.

B. Reducing Energy Density Across Distributed Caches

In conventional allocators [22], memory is allocated at the first available address where a contiguous free block of the requested size exists. Consequently, the allocated memory can reside anywhere within the bounds of the heap and in *nonuniform cache access* multiprocessors in any cache bank within the system. This translates into an increased t_{miss} on L1D misses due to the increased communication distance. *Distance awareness* [23] is a scheme that mitigates this issue by accounting for the communication distance of prospective cache banks while performing memory allocation. However, in doing so, the scheme invariably localizes all data allocated by a PE within a single cache bank, which may already be heavily utilized [24]. This results not only in increased contention but also in nonuniform utilization of available resources, an important contributor to energy density spikes and thermal hotspots. The execution characteristics, memory access characteristics, and mapping of tasks also determine

resource contention and memory access latencies [25]. It is thus essential to consider the actual utilization of caches alongside conventional communication distance when performing memory allocations and account for the individual impact of tasks while mapping them. To this effect, we propose a runtime resource management scheme named *CacheBalancer*, composed of an access-rate-aware memory allocator to homogenize cache bank utilization and a *pain*-aware task mapper to minimize resource contention. Together, the two reduced access latencies within the memory hierarchy ensure uniform utilization of available cache memory resources and, most importantly, decrease energy density (E/A) within the system.

1) Access-Rate-Aware Memory Allocation: Access rate is the fraction of all shared memory references that map to a particular cache bank, and measures the relative utilization of banks within the system. Using two configurable thresholds, this information can be used to effectively guide memory allocation to harmonize utilization

$$\theta_{\text{over}}(p_i, m_j) = \phi \cdot \beta \tag{5}$$

$$\theta_{\text{under}}(p_i, m_j) = \frac{\beta}{\lambda \cdot HC(p_i, m_j)} \tag{6}$$

where θ_{over} and θ_{under} represent the access rate thresholds for a cache bank to be considered as overutliized and underutilized, respectively. β is the midrange value of the access rate across N cache banks in the system and $HC(p_i, m_j)$ is the hop count along the interconnect path between p_i, the PE where the allocation is being performed, and m_j, the candidate cache bank on which the allocated block can reside. The tunable integer parameters ϕ and λ affect allocation flexibility, in terms of the allocator's sensitivity to cache pressure, and tolerance for communication distances, respectively. Since they control the spread of a PE's data across cache banks, their optimal values are functions of cache bank size, miss penalty, interconnect latency, and power budgets. Access rates are measured using counters per bank and are passed through a moving average filter to eliminate temporal spikes that could incorrectly skew the allocator. Filtered values are stored in a globally visible section of the address space, accessible to the allocator. To allocate memory, the allocator function is called from within a task executing on a PE. It first computes the θ_{over} and θ_{under} thresholds using the cache bank access rates and subsequently invokes the selection algorithm listed in Algorithm 1. This yields a target cache bank, within which the requested memory block is allocated.

2) Pain-Aware Task Mapping: Task mapping is performed on the basis of the adapted *pain* metric [26], which describes the performance degradation that would be experienced by already executing tasks if a certain new task were to be mapped onto a particular PE. Pain consists of three parts [45].

1) *Cache Intensity Z(m_j):* Measures how aggressively each cache bank m_j is used by tasks.
2) *Cache Sensitivity S(π_i):* Represents the sensitivity of a task (π_i) to contention in shared cache banks, and is measured as the likelihood of the task's memory hits turning into misses as a result of cache contention.

Algorithm 1 Access Rate Based Cache Bank Selection

```
initial cache bank ← closest cache bank to PE
x ← initial cache bank
Compute θ_over and θ_under
if Access rate[x] > θ_over || Allocated page count[x]> Max. pages per bank then
    x ← next candidate cache bank
    while Access rate[x] > θ_under && Allocated page count[x] > Max. pages per bank do
        x ← next candidate cache bank
        if x==NULL then
            x ← initial cache bank
            break
    if Allocated page count[x]==0 then
        Allocated page count[x] ← number of pages to allocate
        minimum pages ← find minimum non-zero page count
        for every cache bank y do
            Allocated page count[y] ← Allocated page count[y]-min. pages
    else
        Allocated page count[x] ← number of pages to allocate
Target cache bank ← x
Return Target cache bank
```

3) *Communication Impact CI ($\pi_i \rightarrow p_q$):* Encapsulates the degree of performance loss induced due to interconnect contention and communication distance resulting from the mapping of task π_i on PE p_q.
The pain for a task π_i assuming a candidate mapping on PEp_q is thus given by

$$\text{Pain}(\pi_i) = CI(\pi_i \rightarrow p_q) + \sum_{m_j \in M_{\pi_i}} S(\pi_i) \cdot Z(m_j) \tag{7}$$

where M_{π_i} represents the set of caches used by a task π_i. At runtime, the task mapper evaluates the effective pain for each candidate mapping and assigns tasks onto available PEs that yield the lowest pain. In total, the mapping function has a complexity of $O(p)$ per task in a system with p PEs. For n tasks, the complexity is $O(pn \cdot \log(n))$ in the common case ($n \lesssim p$) and $O(pn^2)$ in the worst case ($n \gg p$).

C. Minimizing Latency for Data Transfers

In message-passing architectures, overheads such as t_{mp} and t_{fc} are largely incurred due to the implementation of transfer management functions in software. These functions consume valuable execution cycles in the PE and impact on the effective transfer latencies. To address this, we propose *Pronto*, a message-passing system that integrates transfer management functions into its hardware architecture. This releases PEs from having to explicitly manage data movement and eliminates the need for address translation, synchronization, and resource management to be performed by the programmer. This reduces the overheads incurred in performing message transfers compared with state-of-the-art implementations [16], [17] and allows application communication to be abstracted from its hardware implementation.

Pronto uses a *direct memory access* engine to move data blocks between tile local memories, through hardware-managed *message-passing buffers (MPBs)* over the NoC. Fig. 1(c) illustrates the architecture of a NagaM PE tile

with the Pronto message-passing interface. Data transfers are performed using the *MP_send* and *MP_receive* functions, and calls specify only the size of the message, its location in the tile's local memory, and the sender or recipient's task ID. This provides a high level of abstraction, hiding details such as the actual physical PE onto which tasks are mapped. Each argument of the function maps to a particular control register of the Pronto interface, the contents of which are encoded into a message header (also known as *message envelope*) that precedes each message transfer. This flexible implementation also facilitates the creation of custom message types and other complex primitives.

1) Address Translation: Since the *MP_send* and *MP_receive* function calls only specify task IDs, Pronto relies on a per-tile *address translation table (ATT)* to determine the corresponding physical network address of PEs where the communicating tasks are located. Each ATT is populated by the host during task mapping with the addresses of only the upstream, downstream, and streaming buffer nodes for the task executing on the tile. Their programming overhead is thus negligible and have a fixed single cycle lookup time.

2) Buffer Management: Resource reservations are handled using the *message envelope*, which encodes the physical address of the upstream PE, the message type, and quantum of MPB space required. Envelopes are handled at the downstream node on a first come first served basis, each resulting in the *buffer manager* reserving the requested chunk of memory. MPB reservations and overall utilization are tracked using a status table managed as a circular *FIFO*. Successful reservations result in an acknowledgment to the requesting node (which then commences the transfer), while unsuccessful reservations due to insufficient MPB space result in the corresponding message envelope being buffered. This prevents the need for repeated polling by a waiting upstream node. Consequently, the communication overheads for transfer remains fixed regardless of message size and MPB contention.

3) Ordering of Messages at Destination: The buffer manager preserves the entry order of incoming data blocks using the status table, ensuring that the oldest received block is popped from the buffer when requested by an *MP_receive* call. This guarantees that blocks are consumed in the same order as they are generated.

IV. Thermal-Aware System Design

Design space exploration (DSE) involves the evaluation of architectural and system design options in terms of performance and associated cost. In light of the increased integration densities that accompany stacked-die architectures, thermal behavior assumes greater significance and requires evaluation alongside the traditional metrics of power and performance.

A. Thermal Design Space

Nagata's relation (1) provides a comprehensive formulation of the conflict among integration density, effective activity rate of components, and thermal constraints. This relation assumes greater significance in the case of 3-D ICs due to the complexity of heat flow paths within die stacks. This is evident when (1) is rewritten in the steady-state form of the Fourier heat transfer equation [27]

$$Q = g_{x,y,z} \cdot \Delta T \quad \text{where} \quad g_{x,y,z} = \kappa_{\text{eff}} \frac{A}{l_{x,y,z}} \tag{8}$$

where Q is the quantum of heat flowing from a power dissipating element [represented by the left-hand side of (1)] toward a sink surface of area A through a path of length $l_{x,y,z}$ of a material with effective thermal conductivity κ_{eff}. Thus, $g_{x,y,z}$ defines the effective thermal conductance of the heat flow path between a given power dissipating element and a sink surface, and is dependent on the effective thermal conductivity of all layers that constitute that path. In die stacks, the κ_{eff} of a layer is also influenced by the presence of structures such as TSVs that act as high conductance thermal pathways, improving the conductivity of the layer itself. The κ_{eff} may be computed as the weighted average of the thermal conductivities of the TSV material (κ_{tsv}) and the layer material (κ_{mat}), respectively, as given in (9)

$$\kappa_{\text{eff}} = A_{\text{tsv}}\kappa_{\text{tsv}} + A_{\text{mat}}\kappa_{\text{mat}} \tag{9}$$

$$A_{\text{tsv}} = n(\pi r_{\text{tsv}}^2) \quad \text{and} \quad A_{\text{mat}} = (h_{\text{die}} w_{\text{die}}) - A_{\text{tsv}} \tag{10}$$

where A_{mat} is the area of the material layer in a die with length h_{die} and width w_{die} and A_{tsv} represents the total area occupied by n TSVs of radius r_{tsv} on that layer. The effective thermal conductivity determines the rate at which heat is evacuated away from power dissipating elements, and has a drastic impact on lateral thermal gradients in dies. The location of power dissipation within the stack is also an important factor, influencing the length of the heat flow path and thus its effective thermal conductance. In terms of (8), the magnitude of thermal gradients observed at a given point across any layer of the stack is determined by the rate of heat flow from power dissipating sites to sink surfaces (a function of κ_{eff}, $l_{x,y,z}$, and ΔT) and the power density (Q/A *or* E/At_{pd}) at that point in the stack. These factors are determined by system design choices as well as target process technology parameters, and are central to the concept of *thermal-aware DSE (tDSE)*.

B. Design Framework

Skadron *et al.* [3] proposed one of the earliest tDSE frameworks to explore the interplay between execution and thermal performance. Comprising the *SimpleScalar* architectural simulator [28] and the *Hotspot* thermal model [3], [29], their pioneering framework inspired a number of successors including [30], which detailed a static methodology for thermal-aware performance evaluation of a low-power *multiprocessor system-on-chip (MPSoC)*. Others such as [10] improved the coupling between functional and thermal simulators for tDSE. Most recently, methodologies such as *Pathfinder* [9] enabled the thermal-aware exploration of the 3-D design space. A significant limitation with these methodologies is that they require power and latency models, as well as physical details such as floorplans, vertical interconnect placement, and stack models to be provided as inputs. However, due to the complex nature of the 3-D design space, these inputs are often interdependent. Consequently, a change in one could necessitate modifications to the others

(e.g., change in tile width causes NoC link driver size and power dissipation to change). In a conventional tDSE use case, this highly iterative process is sped up through the use of design abstractions. Despite the effort reduction that it yields, this results in considerable inaccuracies in the simulated thermal behavior [3]. Further, since the internal power density of components is ignored, these methodologies yield overly optimistic performance estimates that deviate drastically from actual physical behavior. Although one framework [31] overcomes this limitation, it relies on full-system synthesis and physical prototyping for accurate characterization, which is time intensive for the purpose of early tDSE.

We developed the *Ctherm* integrated thermal–functional cosimulation framework to overcome this limitation and enable fast and accurate tDSE of 3-D stacked MPSoCs. The framework automates the generation of detailed physical models for components using state-of-the-art *area–latency–energy (ALE)* estimators and exploration methodologies, using only high-level design specifications and abstract floorplans as inputs. This facilitates the accurate characterization of thermal behavior and physical effects corresponding to architecture and system design decisions, early in the flow. Automated generation of component internal floorplans enables accurate characterization of spatial and temporal power dissipation, without requiring significant effort per design point as existing proposals. Since the Ctherm framework contains no proprietary elements, being composed mostly of open-source tooling, it can be readily adopted within other existing EDA tools. It is imperative, however, to ensure that the core principles of Ctherm—accurate spatial and temporal power characterization and accurate physical/thermal modeling—remain undiluted. From our own anecdotal experience, we find that this is not an overly complex endeavor.

The framework consists of two stages: the physical model generator, which translates high-level system specifications into a fine-grained physical model, and the thermal–functional cosimulation platform, which subsequently enables the model to be evaluated. The framework is illustrated in Fig. 2.

1) Physical Model Generation: The physical model comprises fine-grained ALE models for components, generated using parameterized estimators [32]–[34] and translated by a Python interpreter into a standard format. The models contain estimates of dimensions for functional units within components, internal architectural parameters, as well as energy and latency per functional unit operation. The physical model also includes the TSV-based vertical interconnect in order to accurately capture the thermal behavior of the die stack. During TSV topology exploration, candidate vertical interconnect design alternatives are evaluated in terms of electrical performance, implementation cost, and feasibility while accounting for their unique *keep-out zone* requirements [35]. Other design objectives such as whitespace minimization, IR drop, clock-tree size, and temperature [7] can further be supported by the exploration methodology. Fine grained floorplans are subsequently generated by Python-based planning routines, using the component dimensions from the ALE models. These detailed floorplans are inserted into the abstract system-level floorplan, at specific anchor positions,

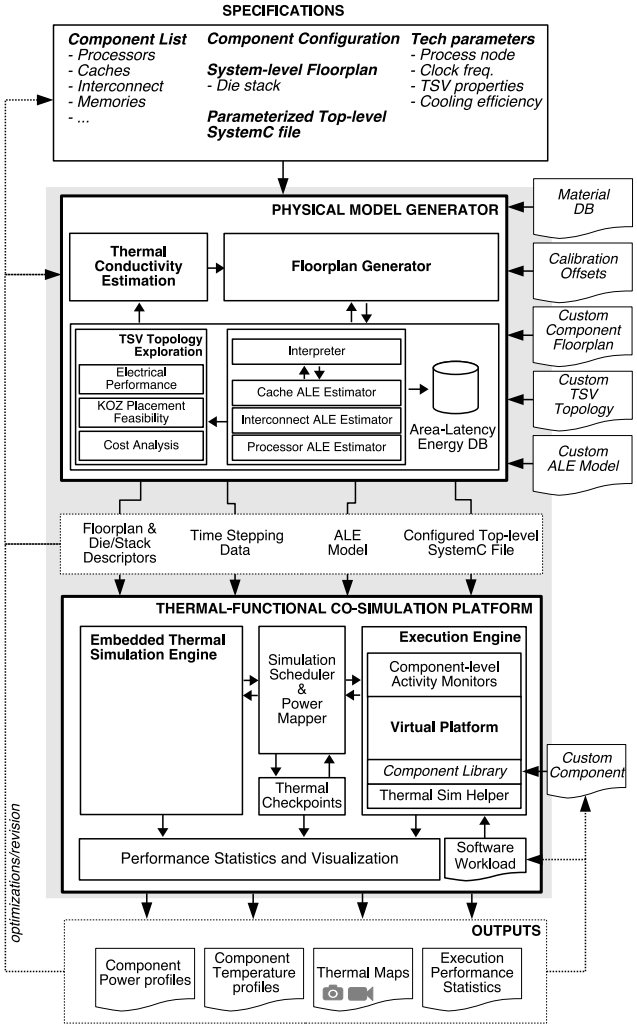

Fig. 2. Ctherm framework for tDSE.

thus enabling the modeling of power dissipation with high spatial resolution and accuracy, with minimal effort. While the internal floorplanner used in Ctherm does not optimize for any specific objective, it can be augmented for temperature and wirelength minimization [8]. For each die in the stack, a descriptor detailing material properties, fine-grained spatial power dissipation, and physical characteristics is generated using a material database. Further, a stack descriptor specifying the stack order, intertier thermal conductivity corresponding to the chosen TSV topology, and cooling interfaces is also generated. This enables accurate characterization of the system's thermal behavior accounting for the parameters E and $\kappa A \Delta T / l$.

2) Thermal–Functional Cosimulation: The second stage of the Ctherm framework performs the thermal–functional evaluation of the MPSoC using the generated physical model and the input SystemC top-level file configured with the system specifications. The cosimulator consists of a cycle-accurate simulation engine with an integrated thermal simulator. The engine instantiates components from the *SoCLiB* IP library [36], which consists of an extensive set of SystemC behavioral models for processor cores, interconnects, caches, memories, controllers, and accelerators. Thermal simulation of modeled platforms is enabled by an adapted version of the

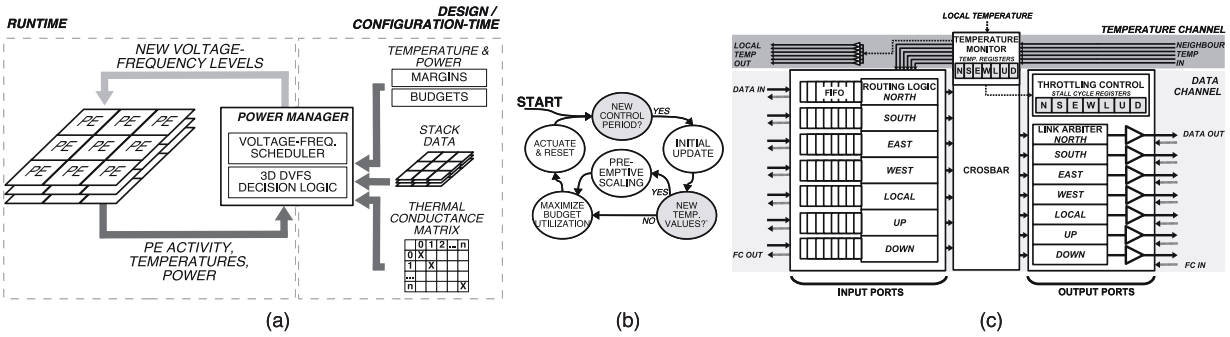

Fig. 3. (a) Illustration of the 3-D DVFS based power manager, and (b) its main states. (c) Illustration of *Immediate Neighbourhood Temperature (INT)* adaptive router architecture with the dedicated temperature channel.

3-D ICE thermal simulation engine [11], embedded within the cosimulation platform core. A thermal model for the system is generated based on the die descriptor and floorplan from the previous stage of the framework, each discretized into a grid of thermal cells. SystemC components of the SoCLiB IP library are augmented with a cycle-accurate activity tracking function that logs the operations performed by their constituent functional within an activity frame. During thermal simulation, frames are evaluated at discrete time steps and converted into detailed power maps using the corresponding ALE models for components. Since the model contains accurate details of the internal organization of components, power mapping is carried out with fine granularity. This is a critical requirement for accurate modeling of thermal behavior, and is necessary for capturing the influence of stack power density on operating temperatures.

V. Runtime Temperature Management

Dynamic thermal managers (DTMs) are used to control on-chip temperatures at runtime, reducing dynamic power dissipation of components by varying activity (α), voltage, and frequency (effectively E/t_{pd}). In critical cases where scaling of voltage and frequency fail to arrest temperatures, measures such as clock and power gating are used to completely freeze activity in circuits, essentially creating a region of inactivity, or *dark silicon*. Consequently, although these strategies afford extensive control over temperature, they result in degraded execution performance. An efficient DTM strategy is one that maximizes system performance within thermal constraints, while concurrently minimizing dark silicon. In a 3-D IC, this implies that the DTM accounts for nonuniformities in temperature margins and the magnitude of thermal coupling arising from the stacking of power dissipating elements.

A. 3-D Dynamic Voltage Frequency Scaling

The operating temperature of a PE is determined by two factors: its thermal coupling with other power dissipating elements in the vicinity and its distance from the sink surfaces. When selecting *voltage–frequency (V-F)* levels for PEs, conventional *dynamic V-F scaling (DVFS)* approaches fail to account for the complex heat flow characteristics of die stacks, and the consequent nonuniformities in temperature margins. This causes relatively cooler PEs close to the heatsink to be overutilized at the cost of their deeply stacked counterparts, resulting in uneven device wear and suboptimal execution performance. Due to the extent of thermal coupling arising from high density integration in 3-D ICs, it is necessary for the DTM strategy to predicate V-F scaling decisions upon their projected thermal impact, i.e., upscaling is prevented from occurring if such an action adversely impacts on another active PE. This forms the core principle of our proposed 3-D DVFS strategy, enabling the utilization of PEs to be maximized within the nonuniform temperature and power budgets in die stacks.

Fig. 3(a) illustrates a power manager implementing 3-D DVFS and its control inputs and outputs. In order to determine appropriate V-F levels for PEs in a stack, 3-D DVFS utilizes runtime monitoring of physical conditions such as activity, power dissipation, and on-chip temperatures. Coupled with a physical model of the die stack indicating the position of PEs and their thermal relationships, DVFS decisions are made using the following algorithm. Fig. 3(b) provides an illustration of its main states.

1) Initialization: In this stage, the internal state of the algorithm is updated with the latest values of available power and temperature budgets for PEs.

2) Preemptive Scaling: The temperature of each PE is protectively maintained within the critical margins by identifying others in the stack that could potentially victimize it due to thermal coupling. Each PE is assigned a weight corresponding to its impact on a victim as

$$\text{weight} = w_a(1-\alpha_i) + w_b \cdot G(victimPE, i) \qquad (11)$$

where w_a and w_b are coefficients whose values are obtained through a linear regression once the physical structure of the stack is known, α_i is the average activity of PE_i, and G represents the normalized thermal conductance matrix for the stack. In the event of a thermal emergency, the PE bearing the highest weight is subject to V-F downscaling, following which the projected temperature impact is estimated as:

$$\delta T_j = \frac{1}{G(i,j)} \alpha_i ({V_{i\ \text{new}}}^2 f_{i\ \text{new}} - {V_{i\ \text{old}}}^2 f_{i\ \text{old}}) \qquad (12)$$

where δT_j is the projected temperature difference arising at PE_j due to scaling of V-F levels from ($V_{i\ \text{old}}$, $f_{i\ \text{old}}$) to ($V_{i\ \text{new}}$, $f_{i\ \text{new}}$) at PE_i. If a scaling action does not sufficiently decrease the temperature of a victim, the process continues iteratively until the aggregate δT is sufficient to bring the victim PE below the critical temperature margin. Clock and power gating serve as the last resort for critical PEs.

3) Maximize Budget Utilization: Temporal variations in execution characteristics of tasks necessitate the continual monitoring of PE activity and budget utilization. In this stage of the algorithm, V-F scaling decisions are fine-tuned to maximize utilization of the available power budget within the prevailing temperature margins. The PEs on which such scaling occurs is determined using the weighted equation

$$\begin{aligned}\text{weight}_{\text{pe}} &= w_c \cdot \alpha_{\text{normalized}} + w_d \cdot \text{temperature margin}_{\text{normalized}} \\ &\quad + w_e \cdot \text{height}_{\text{normalized}} + w_f \cdot \text{area}_{\text{normalized}} \end{aligned} \tag{13}$$

where w_c, w_d, w_e, and w_f are weights whose values are determined through a linear regression at design time and act as multipliers to establish the impact of their corresponding parameters. In this stage, V-F upscaling iteratively targets PEs in decreasing order of their weights, while downscaling targets them in increasing order, until power dissipation reaches the budget value. Note that scaling decisions are followed by a recomputation of projected δT using (12).

4) Actuate and Reset: The new V-F levels as well as control signals for clock and power gating are actuated, and the algorithm is suspended until the next control cycle. The duration of the control period is a function of thermal inertia of the system and the response time of V-F scaling hardware.

B. Thermal-Aware Interconnect

The *NoC* interconnect in multiprocessors consumes a significant fraction of total system power and contributes to the shape of temperature profiles within the system. Modern interconnect architectures integrate a DTM that varies throughput in response to operating temperature [4] or to maintaining power dissipation within the available budget [37]. However, this has the effect of drastically degrading communication performance, especially in the case of 3-D ICs where the nonuniform temperature margins yield persistently skewed network throughputs even at zero load. Thermal-aware adaptive routing is the key to overcoming performance degradation due to throttling DTMs. This also presents an additional opportunity for power control using an interconnect activity to offset load imbalances in the system.

The primary disadvantage with implementing an effective thermal-aware interconnect lies in the cost of broadcasting temperature information across the network. *Traffic and thermal awareness routing (TTAR)* [38] avoids this using the congestion resulting from throttling as an indicator of temperature. *Regional congestion awareness (RCA)* [39] provides the means to aggregate and propagate this congestion information across the interconnect over a dedicated monitoring network. TTAR relies on such an RCA network to determine the location of potential thermal hotspots. However, congestion can also be a consequence of actual interconnect traffic in a relatively low temperature region of the chip, and routing more traffic to this region would only compound the problem. Although using an additional RCA network for temperature provides a workaround, the overheads are prohibitively high.

The physical nature of heat transfer results in thermal hotspots influencing the temperature of other tiles in their vicinity. Therefore, unlike congestion, temperature information does not need to be broadcast across the network in order to facilitate an adaptive thermal-aware routing function. Consequently, as traffic approaches a hotspot, network links in its direction exhibit a higher temperature than those in other directions. Our proposed *immediate neighborhood temperature (INT)* adaptive routing algorithm relies on this observation, steering interconnect traffic based on locally available temperature information alone. It effectively results in the spreading of interconnect activity over a larger area ($\alpha N_G E/A$), thereby reducing energy density.

Algorithm 2 INT Adaptive Routing Algorithm

```
if Address_local == Address_dest then
    Selected Port ← Local
else
    Candidate Ports ← Perform initial OE routing
    Fetch Temperatures of Candidates
    Preferred Candidates ← Candidate port with lowest temp.
    if Preferred Candidates >1 then
        if Up is a Preferred Candidate then
            Selected Port ← Up
        else
            Selected Port ← First Preferred Candidate
    else
        Selected Port ← First Preferred Candidate
Route packet onto Selected Port
```

1) Temperature Monitoring Network: Local temperature at each tile is stored within the *temperature monitor*, from where it is shared with immediate neighboring routers over a dedicated monitoring channel, as shown in Fig. 3c. The area and power overheads incurred by this network are minimal due to the single-hop nonaggregating connections and the fact that temperature updates only occur when new measurements are available from thermal sensors (typically every 50 μs).

2) Adaptive Routing Algorithm: The INT adaptive routing algorithm uses temperature information from neighboring routers to determine the preferred path for packets. The algorithm consists of two steps—an initial routing and a temperature-aware output port selection, as listed in Algorithm 2. In the first stage, the candidate output ports for the packet are determined using the deadlock-free *odd–even (OE)* algorithm. This set of ports encapsulates all the directions in which the packet can egress from the router, making forward progress toward its destination. In the second stage, this set is refined based on operating temperatures, and the waiting packet is routed through the lowest temperature link toward its destination. Although simple, this approach presents an effective low-overhead means of steering interconnect traffic away from high temperatures. The process is self-adjusting as it continually adapts port selection within the constraints of the OE algorithm and prevents overutilization of any single route.

VI. Experimental Results

We demonstrate the efficacy of the proposed techniques in addressing the limitations imposed by dark silicon, using the Naga multiprocessor architecture as the experimental base platform.

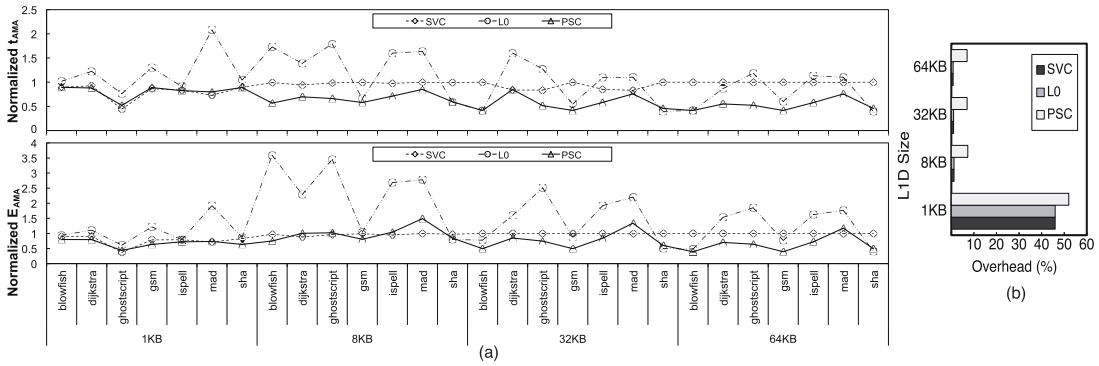

Fig. 4. Comparison of the performance of workloads with a memory hierarchy using *Selective Victim Cache (SVC)*, *L0* and *Persistence Selective Caching (PSC)* assists in terms of (a) average memory access time (t_{AMA}) and average energy per access (E_{AMA}), normalized to a conventional 4-way L1D of size 1KB, 8KB, 32KB and 64KB. (b) Area overhead considering a $11b$ access count and $21b$ window stamp for each line.

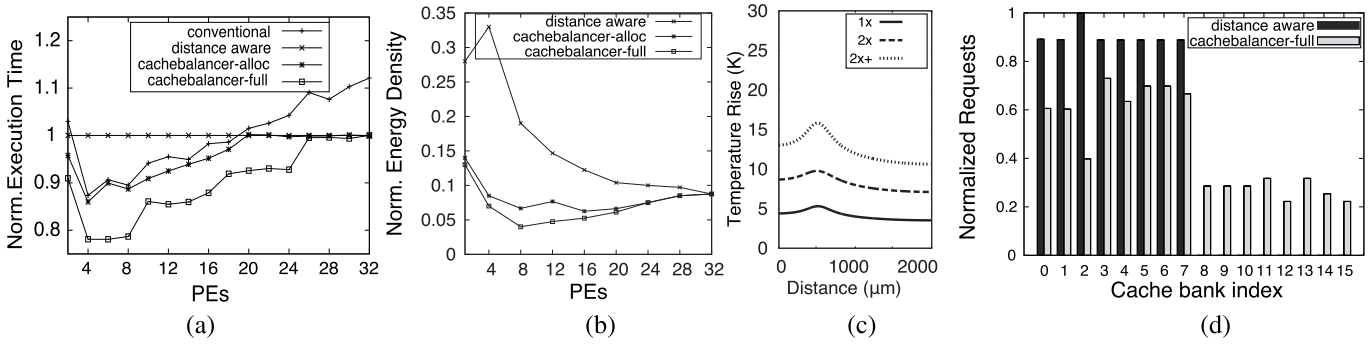

Fig. 5. (a) Execution time for a synthetic workload, normalized to a conventional memory allocator, with 32 PEs and 8 shared cache banks. (b) Normalized energy density. (c) Temperature profiles corresponding to varying energy density ($2\times$ + represents $2\times$ energy density in addition to $1\times$ local dissipation). (d) Illustration of nonuniform cache bank utilization with the distance-aware allocator, in a larger system with 16 banks. Cache configuration L1I: 4 kBytes/2-way/64 Bytes, L1D: 64 kBytes/4-way/64 Bytes, and L2-D: 128 kBytes/8-way/64 Bytes. CacheBalancer uses pressure sensitivity (ϕ): 1 and spread tolerance (λ): 2 hops.

TABLE I

AREA OVERHEADS OVER UNASSISTED DATA CACHE (ASSIST SIZE:512B)

	Latency/Word (cycles)	*Burst Size (words)*
ocMPI [17]	32.9	256
[16]-*Shared Queue*	20	64
[16]-*Scratch Queue DMA*	9	64
Pronto	**6.48**	64

A. Architectural Efficiency

This section investigates the impact of PSC, *CacheBalancer*, and the *Pronto* message-passing system on performance and energy.

1) Persistence Selective Caching: PSC is evaluated using memory traces derived from the in-order execution of workloads from the *MiBench* benchmark suite [40], on a NagaS PE. ALE estimates for the evaluated cases are obtained from Ctherm's integrated estimators. Fig. 5(a) illustrates the average memory access time (t_{AMA}) improvements and energy savings resulting from the use of persistence as a selectivity criterion and from the acceleration of references to reusable lines. The t_{AMA} of the PSC assisted data caches is up to 59% lower than the unassisted case, as well as the competing SVC and L0 schemes across workloads and cache configurations. These improvements are obtained from the selective caching of a relatively small number of reusable cache lines. For the $dijkstra$ workload for a 32-kByte L1D, PSC elevates $8E3$ cache lines to the assist, compared with $20E6$ in the case of the L0. In the notable case of $blowfish$, PSC's t_{AMA} and energy improvements stem from the selective caching of only 29 lines, exemplifying the reusability of data exhibiting sufficient access persistence, and underscoring the impact of accelerating references to such data using the assist. The minimization of t_{AMA} and E_{AMA} together represents an improvement in energy efficiency when considered over the total runtime of the workload. This permits the remaining parameters in (1), activity rate (α), and the number of gates (N_G) to be maximized within the same thermal constraints for the system. The implementation overheads for the PSC assist are compared with the conventional data cache and competing schemes in Fig 4(b),

2) CacheBalancer: Fig. 5 illustrates the execution performance of a synthetic memory heavy workload on a 32-PE NagaS cluster. The workload stresses the memory hierarchy using parallel tasks to repeatedly allocate and write to different memory pages, thus exposing the impact of the allocator and task mapper. Since the workload exploits little locality from the allocated data, it results in frequent L1D misses, causing execution time to be dominated by miss penalty t_{miss}. Fig. 5(a) and (b) reports the execution time and energy density resulting from distance-aware memory allocation [23], CacheBalancer's access-rate-aware memory allocator (*cachebalancer-alloc*), and the full CacheBalancer scheme comprising of the allocator as well as the pain-driven task mapper (*cachebalancer-full*). These results are normalized

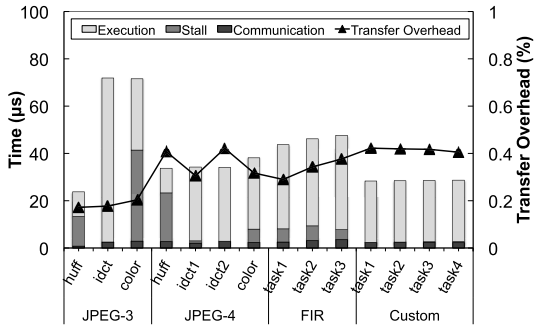

Fig. 6. Decomposed view of task execution on an 18-core 200-MHz NagaM array, with 512-Byte MPBs and 256-Byte output buffers. *Transfer overhead* reflects the overhead imposed by message envelopes as a percentage of total execution time.

TABLE II
COMPARISON OF AVERAGE TRANSFER LATENCY PER WORD

Tech. Node (nm)	90	*Sensor accuracy (K)*	0.5-1
Cooling	Passive	*Sampling interval*	$50\mu s - 1ms$
Heat trans. co-eff. $(Wm^{-2}K^{-1})$	$1E2 - 1E5$	*Voltage (V)*	0.855,0.956,1.048
		Core Freq. (MHz)	0, 700-1200
Temp. range (K)	300-370	*NoC Freq. (MHz)*	500
Critical range (K)	322-332	*Flit width (b)*	38
T cell size (μm)	5-250	*NoC Ports/FIFO*	7/16-deep
Calibration Ref.	[41]	*Packet Size (B)*	4/32/64

to the performance of a conventional multiprocessor memory allocator [22]. The dominant role of communication distance is evident in Fig. 5(a), where all competing allocators yield superior performance compared with the conventional case. However, the drawback of ignoring cache bank utilization results in increased shared cache contention beyond the 18-core mark, overtaking communication distance as the primary bottleneck. This behavior is largely mitigated by homogenizing cache utilization and adapting task mapping to minimize pain, as observed with *cachebalancer-full*. From the perspective of dark-silicon-related thermal challenges, the spreading of utilization across cache banks yields up to 95% lower energy density than the conventional allocator, as observed in Fig. 5(b). The thermal consequences of decreasing power density around a single cache bank are depicted in Fig. 5(c). The homogenized utilization of banks is shown in Fig. 5(d) for a 16-bank system.

3) Pronto: Table II lists the effective zero-load latency measured per word for a burst message transfer between nodes situated at single-hop communication distance from each other using various schemes. The absence of extraneous interconnect traffic at zero-load, single-hop communication distance, and subsaturation injection rates together causes this value to be constituted primarily by the latency of transfer management functions. Integrating transfer management into the message-passing architecture lends Pronto the lowest average transfer latency among the compared proposals. For a 64-word message, Pronto's message envelopes pose an overhead of under 5% relative to the total transfer latency. In the context of real application workloads, this overhead is insignificant. Fig. 6 reports the execution breakdown of three- and four-stage *JPEG* decoders [40], *moving average Finite Impulse Response* filter, and a configurable throughput workload on an 18-PE NagaM cluster. Transfer overheads are measured as the fraction of total execution time spent in performing control functions such as message flow control, resource reservations, and synchronization, which in Pronto amounts to under 0.5%.

B. Thermal-Aware Design

In this section, we demonstrate the thermal-aware design of the Naga multiprocessor through a trajectory that explores the impact of microarchitecture, floorplanning and partitioning, vertical interconnect design, and sensor placement on thermal efficiency and execution performance. Table III lists the parameters of the experimental setup.

1) Microarchitecture Exploration: Microarchitectural design determines the internal organization of components and thus the spatial distribution of power within the system. However, design decisions are typically based solely on delay, area, and power, often resulting in thermal issues passing undetected until late stages in the physical design flow. Left unmitigated, these issues drastically degrade system performance and reliability [2]. Fig. 7(a)–(d) illustrates the thermal behavior of three 32-kByte data caches with identical configurations, varying only in their internal microarchitecture. The N_{spd} parameter determines the number of cache lines stored within each sub-bank word line, influencing the distribution of access energy across the SRAM array (spatial power density). Fig. 8 depicts the thermal behavior of the cache following $1E5$ sustained accesses to a single cache line. Fig. 7(a) depicts the thermal behavior with abstracted internal microarchitecture details, typical of conventional tDSE approaches in the literature. The temperature inaccuracy stemming from this abstraction can be as high as 70%, compared with Ctherm's accurate characterization in Fig. 7(b)–(d). Thus, while a cache designer may vary the internal microarchitecture of a cache in order achieve a particular power performance target, the resulting thermal effects of such a configuration may actually be detrimental to system performance, as is observed by the magnitude of the hotspot in Fig. 7(b). This is also the case for architectural design choices that impact on temporal power density. For instance, with schemes such as PSC, a decrease in *cycles per instruction (CPI)* results in faster execution, which, without a commensurate energy benefit, can result in the rapid increase in temperature and cause hotspots to form. Fig. 7(e) depicts this behavior for the *first sum (kernel11)* workload from the *Livermore Loop Kernels* benchmark [42] with an array size of 168×168. PSC yields a 24% reduction in CPI, but also results in a higher temperature rise. However, due to the associated reduction in energy per access, the final temperature is still 1 K lower than the unassisted case.

2) Floorplanning and Partitioning: Fig.9(a)–(b) report the impact of system floorplanning and partitioning on effective execution performance, using a small subset of the NagaS array. Cases FP_A and FP_B present alternative placement strategies for PEs within tiles, while FP_C and FP_D further partition the system into multiple dies for stacking. Although these floorplans vary spatial power density, they also affect interconnect power dissipation due to their influence on link length and thus driver size.

The combination of decreased power density and driver size lends FP_B the lowest operating temperature and 10% fewer execution stalls due to the clock-gating DTM.

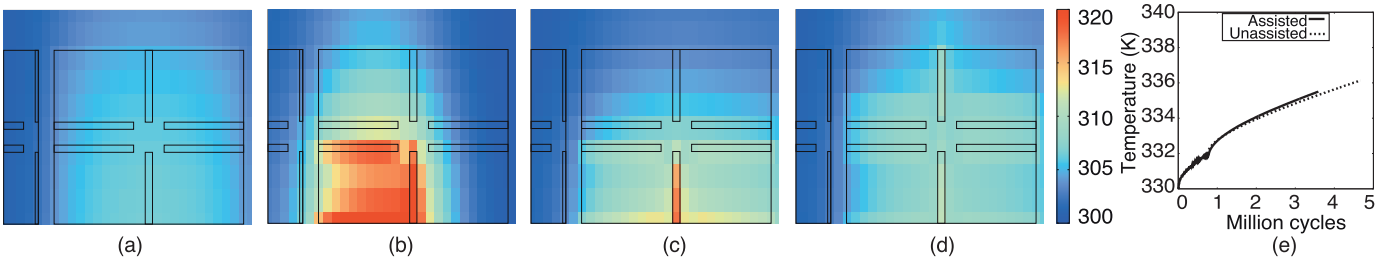

Fig. 7. Access power distribution across sub-banks in a cache data array with varying N_{spd} with corresponding heatmaps, for 32KB data cache using (a) conventional state-of-the-art thermal simulation with abstracted cache internals, and Ctherm with fine-grained cache floorplans for (b) $N_{spd} > 1$, (c) $N_{\text{spd}} = 1$, (d) $N_{\text{spd}} < 1$. (e) Runtime thermal performance comparison of unassisted and assisted data cache. All temperatures are in *Kelvin (K)*.

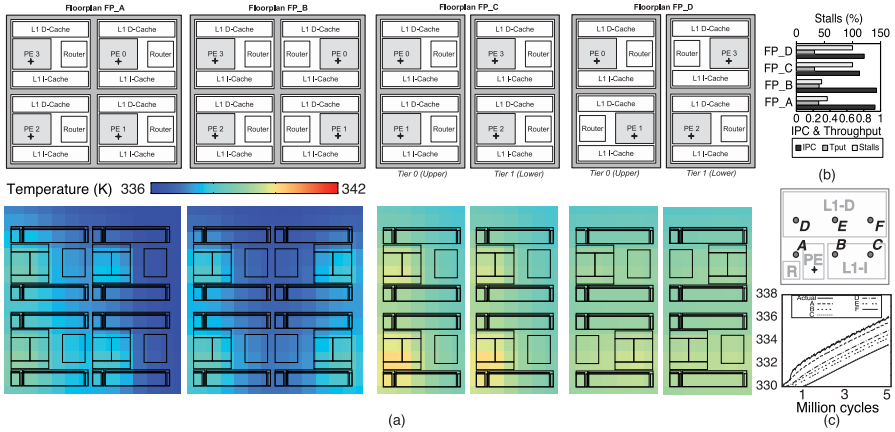

Fig. 8. (a) Floorplan options for a subset of a NagaS cluster, with corresponding thermal maps for execution of the *dijkstra* workload. (b) Execution performance for each, considering temperature sensors placed at locations marked by +. (c) Accuracy of temperature measurements from different candidate sensor locations. Temperatures are measured in *Kelvin (K)*.

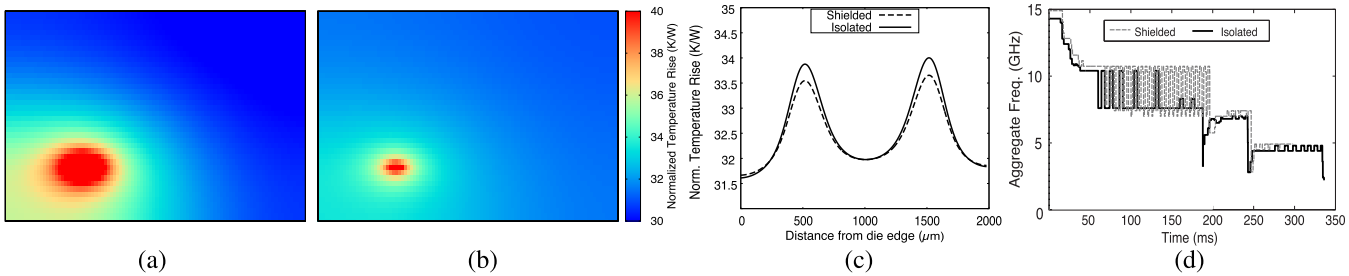

Fig. 9. Thermal map depicting normalized temperature rise as κ_{eff} is increased from (a) 1× to (b) 10× due to TSVs. (c) Distribution of temperature rise per unit dissipation of power at two sites located at 500 and 1500μm with the shielded and isolated TSV topologies. (d) Corresponding aggregate frequency profile of execution on a three-tier 12 PE NagaS cluster with a conventional DVFS power manager, with the two topologies.

FP_C and FP_D rely on TSV-based vertical links, which are 10× shorter than planar wiring. However, despite the consequent benefits that stem from reduced link driver size, these floorplans double power density without improving cooling efficiency. Although FP_D mitigates this to a small extent with its checkerboard placement pattern, the net execution performance is still inferior to FP_A and FP_B. More importantly, FP_C and FP_D result in the system entering the thermal runaway state, despite PEs being clock gated by the DTM. This highlights the need for leakage control and power gating mechanisms especially in lower tiers when high-efficiency heatsinks and forced cooling cannot be used.

3) Vertical Interconnect Design: The choice of TSV topology impacts the shape of temperature profiles within stacked dies. Fig. 9 illustrates the rapidly diminishing size and magnitude of a steady-state hotspot as effective thermal conductivity of the die is increased by the addition of TSVs. Topology choices are typically motivated by the electrical performance and nontrivial area overheads of TSVs [35]. For instance, the *shielded* topology intersperses signal nets with shield TSVs connected to power/ground rails as a means of decreasing coupling induced noise. On the other hand, the *isolated* topology spaces signal TSVs for the same reason, with a decreased area overhead. Despite achieving similar electrical performance, the two differ in thermal performance. Notably, the shielded topology offers a slightly higher operating temperature margin due to its superior κ_{eff}. Fig. 9(c) depicts this increased margin in the single-axis temperature profile of a die stacked above two power dissipating elements. Although the additional margin amounts to less than 0.5 K/W, Fig. 9(d) illustrates the considerable effect this has on execution performance of a three-tier 12 PE NagaS cluster with a conventional DVFS

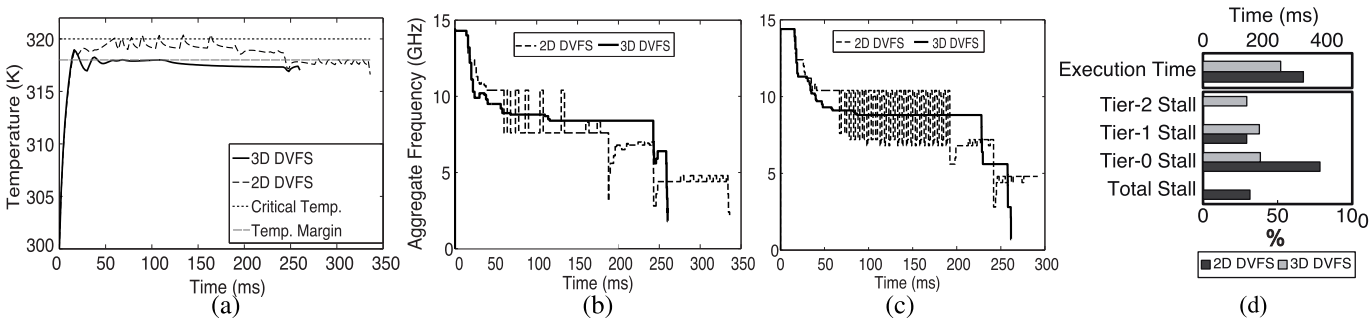

Fig. 10. (a) Temperature profiles for a PE on the lowest tier using isolated TSV topology. (b) Corresponding aggregate system frequency. (c) Aggregate frequency with shielded topology. Both DVFS versions support three supply voltages (0.855, 0.956, and 1.048 V), and two frequencies for each voltage level (700/800 MHz, 900/1000 MHz, and 1100/1200 MHz) in the 90-nm node.

power manager [43]. The additional margin enables the system to operate at a higher V-F level and results in execution completing 38 ms sooner than in the case of the isolated topology. This 11% speedup is simply obtained from the choice of TSV placement topology.

4) Temperature Sensor Placement: Modern temperature sensors incorporating an *analog-to-digital converter* are large and often cannot be placed in close proximity to the component requiring thermal monitoring. This induces inaccuracies in measurements and compensating for them requires characterization of spatiotemporal thermal behavior, a step conventionally performed at the end of physical design or even post silicon. Fig. 8(c) illustrates how calibration offsets can be derived based on sensor placement (locations A–F) around a power dissipating PE, to compensate for inaccuracies in readouts.

C. Runtime Management

The proposed 3-D DVFS and INT routing algorithm are evaluated in terms of their ability to maintain operating temperatures below critical levels, the peak magnitude of thermal gradients within the system, and the effective performance resulting from their use. In the case of DVFS, this is measured in terms of execution time, while for the interconnect, this consists of transfer latencies.

1) 3-D DVFS: The DVFS power manager is evaluated using a cluster of 12 NagaS PEs in a three-tier stack, with the multiprogrammed execution of the *basicmath* workload [40].

Fig. 10(a) illustrates the temperature of a PE farthest from the heatsink, situated on the lowest tier of the three-tier isolated TSV-topology-based stack with both conventional 2-D DVFS [43] as well as the proposed 3-D DVFS. Since conventional 2-D DVFS is oblivious to the physical structure of the stack and considers only the local temperature of PEs when performing V-F scaling, and it exhibits inadequate control over operating temperatures. Consequently, frequent fluctuations and breaches of the critical limit are observed, resulting in stalled execution and degraded performance. On the other hand, since 3-D DVFS predicates scaling upon the projected impact it would have on other PEs in the system, temperatures are held well below critical. Furthermore, since operating temperatures are lowered, effective utilization (α) is increased by preventing performance degrading execution stalls from occurring, as evidenced in Table V. Aggregate frequency is consequently higher, since more PEs are active at any given time. Execution therefore completes at least 19% sooner compared with 2-D DVFS, for the same workload. In addition, utilization is homogenized across all tiers, as opposed to only the cooler tiers close to the heatsink. Essentially, 3-D DVFS balances the activity distribution ($\alpha N_G E/t_{\text{pd}}$) within the constraints imposed by the system's external cooling interfaces, while mitigating the impact of internal nonuniformities by managing the system as a single cohesive unit. This latter observation is evidenced in Fig. 10(b) and (c), which compares the equivalent system performance with the isolated and shielded TSV topologies. Notably, 3-D DVFS yields the same execution performance in both cases, which represents the optimum for the given system's externally constrained thermal operating envelope. This observation illustrates the nontriviality of the 3-D design space, where the choice of power management algorithm can influence TSV topology decisions.

2) INT Adaptive Routing: INT is evaluated using *Ctherm* with a 64-PE ($4 \times 4 \times 4$) *NagaM* multiprocessor cluster. Fig. 11 characterizes the throttling response of the INT-based routers for each 0.5 K rise in temperature and the impact on IPC of PEs. For dataflow multiprocessors, this demonstrates how the interconnect can regulate activity in PEs and thus control power dissipation. The performance in balancing thermal gradients and improving network latencies is evaluated using synthetic traffic patterns. INT is benchmarked against *downward routing* [44], which preferentially utilizes tiers close to the heatsink to route packets, *TTAR* [38], and *TTAR+*, which adds a dedicated temperature monitoring RCA network [39]. Fig. 12 compares the temperature maps for all four tiers of the stack obtained with INT and TTAR, in the presence of eight interconnect extraneous thermal hotspots. This emulates load imbalances in the system due to executing workloads that cannot be throttled by the interconnect.

The superiority of INT's thermal-aware routing strategy is observed in its smoother profiles and lower operating temperatures compared with that of TTAR. This results in reduced throttling, thereby decreasing latency and congestion across the entire stack, as observed in Fig. 12(b). Further, INT does not preferentially utilize any specific tier, as opposed to other schemes. Interestingly, its ability to regulate thermal gradients

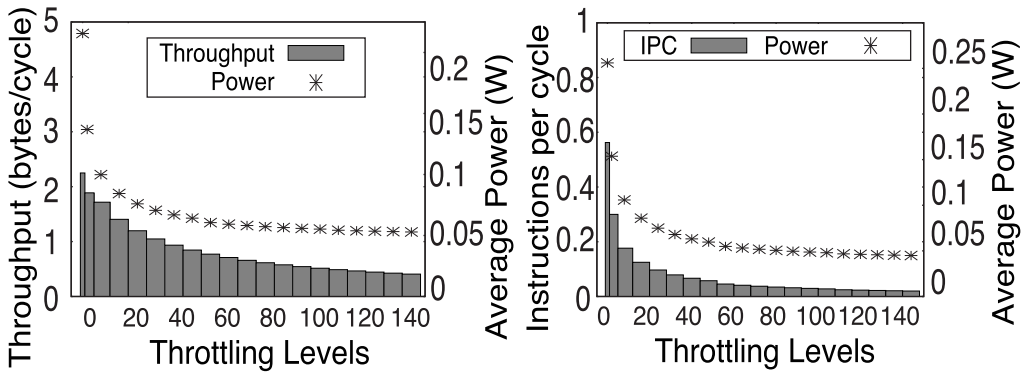

Fig. 11. Influence of throttling on throughput and effective IPC of NagaM PE as well as power dissipation.

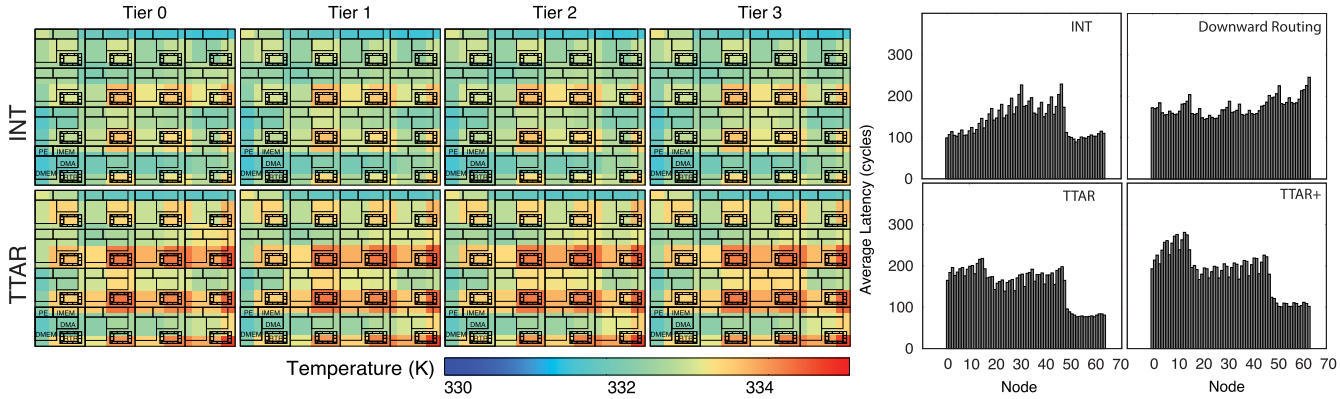

Fig. 12. Temperature maps from *uniform random* traffic with eight thermal hotspots with different routing strategies and the corresponding average latency distribution across the network after 750 kcycles.

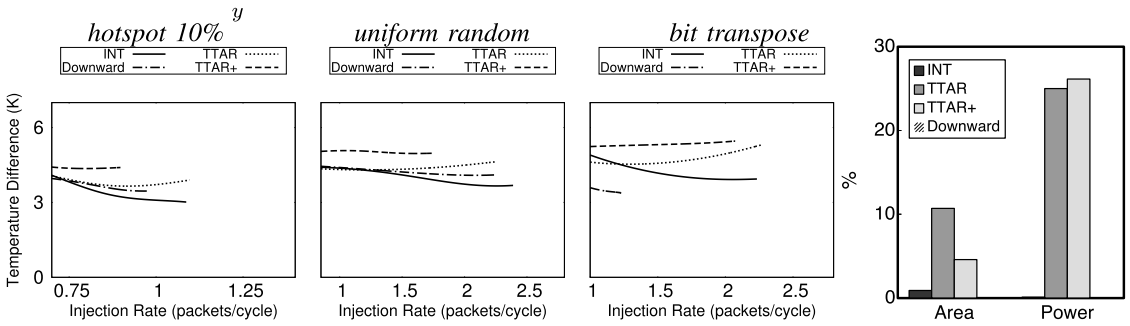

Fig. 13. Peak temperature difference with eight thermal hotspots for different traffic patterns. The data lines end at different injection rates corresponding to the saturation throughput of the network.

is predicated upon the existence of sufficient traffic within the interconnect, as seen in Fig. 13. However, the effective impact of thermal-aware routing schemes is limited by the overheads incurred in operating global temperature monitoring networks. The thermal degradation induced due to this additional power dissipation can even overshadow the benefits bestowed by the adaptive routing approach, "as observed in the case of TTAR+ ". Due to INT's minimal temperature monitoring network, power overheads are minimal, as observed in Fig. 14 (TTAR+ uses 5-bit temperature and 3-bit temperature channels, respectively).

VII. Conclusion

This paper examines the critical causes of dark silicon and details a comprehensive trajectory that addresses these at the architecture level in the system design flow and at runtime. First, the paper highlights the impact of the memory hierarchy and interconnect on system performance and focuses on improving energy efficiency as a means of mitigating power density issues. Second, this paper presents a novel tDSE flow to practically evaluate the complex thermal implications of system design decisions. Finally, this paper describes runtime strategies to minimize the impact of dark silicon by adapting system behavior to fit specific thermal constraints. Together, these provide a sense of the critical issues that designers need to actively address, in order to successfully realize dependable thermal-aware 3-D systems.

References

[1] M. Nagata, "Limitations, innovations, and challenges of circuits and devices into a half micrometer and beyond," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 465–472, Apr. 1992.

[2] K. Ramakrishnan, R. Rajaraman, S. Suresh, N. Vijaykrishnan, Y. Xie, and M. J. Irwin, "Variation impact on SER of combinational circuits," in *Proc. 8th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2007, pp. 911–916.

[3] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, 2004.

[4] K. Puttaswamy and G. H. Loh, "Thermal herding: Microarchitecture techniques for controlling hotspots in high-performance 3D-integrated processors," in *Proc. IEEE 13th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2007, pp. 193–204.

[5] A. K. Coskun, A. B. Kahng, and T. S. Rosing, "Temperature- and cost-aware design of 3D multiprocessor architectures," in *Proc. 12th Euromicro Conf. Digit. Syst. Design, Archit., Methods Tools (DSD)*, Aug. 2009, pp. 183–190.

[6] S. S. Kumar, A. Aggarwal, R. S. Jagtap, A. Zjajo, and R. van Leuken, "System level methodology for interconnect aware and temperature constrained power management of 3-D MP-SOCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 7, pp. 1606–1619, Jul. 2014.

[7] J. Knechtel, I. L. Markov, J. Lienig, and M. Thiele, "Multiobjective optimization of deadspace, a critical resource for 3D-IC integration," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2012, pp. 705–712.

[8] D. Cuesta, J. L. Risco-Martin, J. L. Ayala, and D. Atienza, "3D thermal-aware floorplanner for many-core single-chip systems," in *Proc. 12th Latin Amer. Test Workshop (LATW)*, Mar. 2011, pp. 1–6.

[9] S. Priyadarshi, W. R. Davis, M. B. Steer, and P. D. Franzon, "Thermal pathfinding for 3-D ICs," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 4, no. 7, pp. 1159–1168, Jul. 2013.

[10] A. Bartolini, M. Cacciari, A. Tilli, L. Benini, and M. Gries, "A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores," in *Proc. 20th Symp. Great Lakes Symp. VLSI* , May 2010, pp. 311–316.

[11] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2010, pp. 463–470.

[12] R. Hameed *et al.*, "Understanding sources of inefficiency in general-purpose chips," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, Jun. 2010, pp. 37–47.

[13] B. Bohnenstiehl *et al.*, "A 5.8 pJ/Op 115 billion ops/sec, to 1.78 trillion ops/sec 32nm 1000-processor array," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Jun. 2016, pp. 1–3.

[14] S. S. Kumar and R. van Leuken, "A 3D network-on-chip for stacked-die transactional chip multiprocessors using through silicon vias," in *Proc. 6th Int. Conf. Design Technol. Integr. Syst. Nanosc. Era (DTIS)*, Apr. 2011, pp. 1–6.

[15] M. P. Forum, "MPI: A message-passing interface standard," Univ. Tennessee, Knoxville, TN, USA, Tech. Rep., 1994.

[16] P. Francesco, P. Antonio, and P. Marchal, "Flexible hardware/software support for message passing on a distributed shared memory architecture," in *Proc. Design, Autom. Test Eur.*, Mar. 2005, pp. 736–741.

[17] J. Joven, F. Angiolini, D. Castells-Rufas, G. De Micheli, and J. Carrabina, "QoS-ocMPI: QoS-aware on-chip message passing library for NoC," in *Proc. Workshop Program. Models Emerg. Archit.*, 2010.

[18] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," *SIGARCH Comput. Archit. News*, vol. 18, no. 2SI, pp. 364–373, May 1990.

[19] D. Stiliadis and A. Varma, "Selective victim caching: A method to improve the performance of direct-mapped caches," *IEEE Trans. Comput.*, vol. 46, no. 5, pp. 603–610, May 1997.

[20] A. Janapsatya, S. Parameswaran, and A. Ignjatović, "HitME: Low power Hit MEmory buffer for embedded systems," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2009, pp. 335–340.

[21] N. Duong, T. Kim, D. Zhao, and A. V. Veidenbaum, "Revisiting level-0 caches in embedded processors," in *Proc. Int. Conf. Compilers, Archit. Synthesis Embedded Syst.*, Oct. 2012, pp. 171–180.

[22] E. D. Berger, K. S. McKinley, R. D. Blumofe, and P. R. Wilson, "Hoard: A scalable memory allocator for multithreaded applications," *ACM SIGPLAN Notices*, vol. 35, no. 11, pp. 117–128, Nov. 2000.

[23] S. Cho and L. Jin, "Managing distributed, shared L2 caches through OS-level page allocation," in *Proc. 39th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2006, pp. 455–468.

[24] A. Ros, M. Cintra, M. E. Acacio, and J. M. García, "Distance-aware round-robin mapping for large NUCA caches," in *Proc. Int. Conf. High Perform. Comput. (HiPC)*, Dec. 2009, pp. 79–88.

[25] T. Agarwal, A. Sharma, A. Laxmikant, and L. V. Kale, "Topology-aware task mapping for reducing communication contention on large parallel machines," in *Proc. 20th Int. Parallel Distributed Process. Symp. (IPDPS)*, Apr. 2006, p. 10.

[26] S. Zhuravlev, S. Blagodurov, and A. Fedorova, "Addressing shared resource contention in multicore processors via scheduling," *SIGPLAN Notices*, vol. 45, no. 3, pp. 129–142, Mar. 2010.

[27] A. Zjajo, N. van der Meijs, and R. van Leuken, "Dynamic thermal estimation methodology for high-performance 3-D MPSoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 9, pp. 1920–1933, Sep. 2014.

[28] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, Feb. 2002.

[29] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.

[30] G. Paci, F. Poletti, L. Benini, and P. Marchal, "Exploring 'temperature-aware' design in low-power MPSoCs," in *Proc. Conf. Design, Autom. Test Eur.*, Mar. 2006, pp. 838–843.

[31] D. Milojevic *et al.*, "Automated pathfinding tool chain for 3D-stacked integrated circuits: Practical case study," in *Proc. IEEE Int. Conf. 3D Syst. Integr. (3DIC)*, Sep. 2009, pp. 1–6.

[32] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2007, pp. 3–14.

[33] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. 27th Annu. Int. Symp. Comput. Archit.*, 2000, pp. 83–94.

[34] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration," in *Proc. Conf. Design, Autom. Test Eur.*, Apr. 2009, pp. 423–428.

[35] R. Jagtap, S. S. Kumar, and R. V. Leuken, "A methodology for early exploration of TSV placement topologies in 3D stacked ICs," in *Proc. 15th Euromicro Conf. Digit. Syst. Design (DSD)*, Sep. 2012, pp. 382–388.

[36] SoCLiB-Project. *Soclib: An Open Platform for Virtual Prototyping of Multi-Processors System on Chip*, accessed on Dec. 1, 2014. [Online]. Available: http://www.soclib.fr

[37] L. Shang, L.-S. Peh, and N. K. Jha, "PowerHerd: A distributed scheme for dynamically satisfying peak-power constraints in interconnection networks," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 1, pp. 92–110, Jan. 2006.

[38] S.-Y. Lin et al, "Traffic- and thermal-aware routing for throttled three-dimensional network-on-chip systems," in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, Apr. 2011, pp. 1–4.

[39] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proc. IEEE 14th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2008, pp. 203–214.

[40] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. IEEE Int. Workshop Workload Characterization (WWC)*, Dec. 2001, pp. 3–14.

[41] H. Oprins *et al.*, "Fine grain thermal modeling and experimental validation of 3D-ICs," *Microelectron. J.*, vol. 42, no. 4, pp. 572–578, Apr. 2011.

[42] F. Mahon, *The Livermore Fortran Kernels: A Computer Test of the Numerical Performance Range*. Livermore, CA, USA: Lawrence Livermore National Laboratory, 1986.

[43] X. Wang, K. Ma, and Y. Wang, "Adaptive power control with online model estimation for chip multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 10, pp. 1681–1696, Oct. 2011.

[44] C.-H. Chao, K.-Y. Jheng, H.-Y. Wang, J.-C. Wu, and A.-Y. Wu, "Traffic- and thermal-aware run-time thermal management scheme for 3D NoC systems," in *Proc. 4th ACM/IEEE Int. Symp. Netw.-Chip (NOCS)*, May 2010, pp. 223–230.

[45] J. de Klerk, S. S Kumar, and R. van Leuken, "Cache balancer: Access rate and pain based resource management for chip multiprocessors," in *Proc. Int. Symp. Comput. Netw.*, 2014, pp. 453–456.