2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

Ctherm: An Integrated Framework for Thermal-Functional Co-Simulation of Systems-on-Chip

Sumeet S. Kumar, Amir Zjajo, Rene van Leuken
Circuits and Systems Group, Faculty of EEMCS,
Delft University of Technology, The Netherlands
{s.s.kumar, a.zjajo, t.g.r.m.vanleuken}@tudelft.nl

***Abstract*—This paper presents Ctherm, an integrated framework for cycle-accurate thermal and functional evaluation of systems-on-chip. The presented framework enables accurate characterization of thermal behaviour by generating detailed physical models for components based on input specifications, and simulating them within a tightly integrated co-simulation platform with an embedded thermal simulator. Ctherm's fine-grained modelling approach yields 70% higher accuracy in hotspot resolution as compared to conventional approaches that abstract component internals. Simulation runtime time is reduced by upto 36% over conventional continuous approaches through the use of thermal checkpointing, enabling the fast-forwarding of thermal simulations without loss of thermal continuity.**

I. Introduction

The increasing integration densities of modern high performance *systems-on-chip (SoC)* has resulted in issues such as thermal hotspots becoming a common occurrence. Conventional system-level *design space exploration (DSE)* frameworks, however, do not include such effects into their evaluation of design options [1]. This leads to actual system performance varying significantly from DSE estimates, and yielding thermally inefficient designs. Early knowledge of runtime thermal behaviour of systems can be invaluable in guiding architectural and system-level design decisions. A *thermal-aware DSE (tDSE)* approach can enable the evaluation of execution performance and thermal cost of architectural choices, for instance, planar versus 3D stacked-die implementations for a *multiprocessor SoC (MPSoC)*.

Over the years, a number of methodologies, frameworks and tool flows have been proposed to meet this need. Skadron et al., proposed one of the earliest thermal-functional co-simulation frameworks comprising of the *SimpleScalar* architectural simulator and the *Hotspot* thermal model [2]. Their pioneering work was followed by a number of other proposals such as [3], which integrates a multi-core processor simulator with a static thermal model to enable thermal-aware performance evaluation of a low power MPSoC. Proposals like [4] and [5] improved the coupling between functional and thermal simulators to enable modelling of *Dynamic Thermal Management (DTM)* schemes, and others such as [6] and [7] enabled the thermal-exploration of system-designs using *SystemC/TLM* models. The growing popularity of 3D design for SoCs has led to the development of methodologies such as *Pathfinder3D* [8] and *PathfindingFlow* [9] for the thermal-aware exploration of the 3D design space, and the *MEVA-3D* [10] floorplanner for physical design and performance estimation of 3D microarchitectures.

Despite their merits, all the surveyed proposals exhibit a number of limitations that motivate this paper. Firstly, the existing proposals require power and latency models to be provided as an input along with system floorplans. Since such methodologies are most often used in tDSE scenarios, abstractions are used to avoid the time-consuming development of detailed models and component floorplans for each configuration at each design point. While this enables the fast and simple revision of system specifications, it leads to inaccuracies in thermal characterization [2]. Secondly, existing methodologies do not model the internal organization of components such as caches, resulting in abstractions that hide the internal power dissipation characteristics of individual components. The ability to accurately characterize component thermal behaviour is a prerequisite to developing thermally efficient systems. Thirdly, existing methodologies require the thermal simulation to be repeated from scratch even for minor modifications and optimizations to the system, eg. a change in the DTM's critical temperature.

These limitations of existing proposals motivates *Ctherm*, which is an integrated co-simulation framework that enables thermal-aware design of systems on chip. It simplifies the design process by automating the generation of detailed floorplans and *area-latency-energy (ALE)* models for components from the input system specification. Abstract components in the system-level floorplans are replaced with the generated floorplans, preserving the flexibility afforded by abstraction while removing the inaccuracy associated with their use. The modeled system is instantiated as a SystemC virtual platform together with an embedded thermal engine in the cycle-accurate co-simulation stage to estimate functional as well as thermal performance.

The primary contributions of this paper are:

- An integrated framework for thermal-functional co-simulation of 2D/3D SoCs with automated generation of detailed floorplans and area-latency-energy models for system components.
- Enables fine-grained thermal evaluation of systems,

1066-6192/15 $31.00 © 2015 IEEE
DOI 10.1109/PDP.2015.56

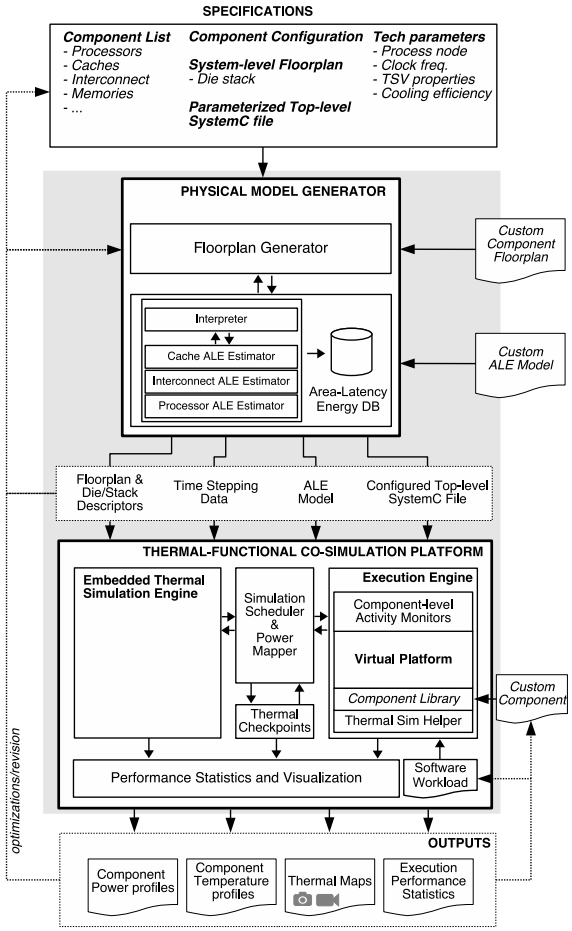

Fig. 1. Ctherm framework for thermal-functional co-simulation

and allows 70% higher accuracy for thermal characterization of components such as cache memories by including their internal organization in the system-level floorplan.

- Reduces the simulation time for iterative post-optimization runs by upto 36% through the use of reloadable thermal checkpoints.
- Illustrates the potential of cycle-accurate thermal aware design using four realistic design cases.

II. Ctherm Framework

The Ctherm framework consists of two stages: the physical model generator, and the thermal-functional co-simulation platform. These stages enable the translation of input specifications into a physical model of the system, and subsequently, the thermal-functional evaluation of the model to determine system performance and thermal efficiency. The Ctherm framework is illustrated in Figure 1.

A. Physical Model Generation

The physical model consists primarily of a system-level floorplan and *area-latency-energy (ALE)* models for components. We generate the physical model in two stages; first, by estimating the latency, energy and dimensions for components based on their configuration, and second, generating fine-grained floorplans for individual components based on these estimates, and inserting them into the system-level floorplan.

1.) Area-Latency-Energy Model: The thermal behaviour of components is largely dependent on their internal organization, power dissipation characteristics and area. Accurate characterization requires detailed models of the energy and latency per operation of each component, together with the area of its constituent functional units. ALE data for generic components such as cache memories, interconnects and simple processor cores can be generated using existing parameterizable estimators [11][12][13]. Since SoCs are often composed of such generic components, Ctherm integrates a number of state-of-the-art estimators within its configuration generator. A Python interpreter is used to translate input system specifications into a suitable format for each estimator, and further convert the outputs of each into usable ALE data. ALE models contain estimates of dimensions for components and their constituent functional units, architectural organization (such as number of cache banks, interconnect ports), as well as energy and latency per operation, and per functional unit.

i. Cache ALE estimation: The internal organization of on-chip cache memories often vary based on the configuration (size, associativity and cache line size) as well as the chosen design target (minimized area, latency and power dissipation). For instance, optimizations such as wordline and bitline segmentation that are used to reduce access energy also result in large *SRAM* arrays being divided into multiple smaller sub-banks, each with its own decoder, sense amplifier, comparators and output drivers. This change in organization can significantly affect how dissipated power is distributed across the the entire cache area. Thus it is essential to include such estimates into ALE models in order to obtain an accurate thermal characterization of components. A summary of data included within cache ALE models is listed in Figure 2.

The *wordline sharing factor (N_{spd})* parameter best illustrates the effect of changing component internal organization on power distribution. This factor specifies the number of cachelines stored per *SRAM* wordline, and is used to optimize caches to meet specific power, area or latency constraints. N_{spd} essentially also determines which SRAM banks are activated on memory accesses. For instance, a value greater than one results in the same SRAM wordline line being activated for

func. unit dimensions	bitline segmentation	access latency
# decoders	wordline segmentation	cache size
# output drivers	sense amp/driver mux	associativity
# comparators	sharing factor (N_{spd})	line size
# sense amplifiers	energy per func. unit	

Fig. 2. ALE data for cache memories

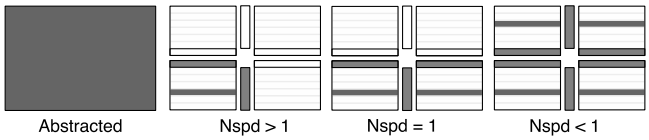

Fig. 3. Access power distribution across sub-banks in a cache data array with varying N_{spd}

# ports	# TSV
driver energy	TSV topology
crossbar energy	TSV conductivity
input port energy	func. unit dimensions
arbiter energy	

Fig. 4. ALE data for interconnect routers

accesses to all cache lines that share it, and conversely for N_{spd} values lesser than one, multiple SRAM wordlines being activated for accesses to a single cache line. The location of the dissipated power for accesses is observed to be strongly linked to the N_{spd} value, and in cases can result in the formation of thermal hotspots. Ignoring this parameter during thermal simulation results in the power dissipated on accesses to be distributed across the entire cache's area, as illustrated by the Abstracted case in Figure 3.

ii. Interconnect ALE estimation: Network-on-Chip interconnect routers consist of multiple input/output ports arranged around a central crossbar switch. The area, latency and energy of routers is influenced by the width of the NoC links, their physical length (output driver size), and the number of ports. For 3D routers, Through Silicon Via (TSV) count and topology are also important parameters as they determine the thermal conductivity between tiers of the die-stack. Interconnect ALE estimation for 3D routers must also determine the optimal TSV topology, and thus total TSV count per switch. In Ctherm, this is done using an exploration methodology that determines the optimal TSV topology based on placement feasibility, electrical performance and area cost. The details of this methodology are covered in our earlier paper [14]. Figure 4 lists the significant data fields within the router ALE model.

iii. Processor ALE estimation: The ALE model for processors can be generated in one of two ways - per instruction, and per pipeline stage. The former lumps the power of individual pipeline stages into an aggregate value for each instruction, while the latter specifies a generalized average power value per pipeline stage regardless of the actual instruction. Processor ALE models consist primarily of functional unit dimensions, and the power dissipation at the chosen granularity.

For other components, custom ALE estimators can be added to the framework by simply extending the Python interpreter. However, for components with optimized implementations, and those for which no parametrizable estimators exist, custom ALE models must be provided as an input to the configuration generator.

2.) Floorplan Generation: To enable fine-grained thermal characterization of systems, Ctherm automates the generation of floorplans based on component dimensions extracted from ALE models, using specific Python based planning routines for each component type. These routines only require specification of the anchor position for components in the system floorplan. The pseudo-code for the planning routines used for memories, interconnect routers and processing elements is listed in Figure 5. For components without a rigid internal organization, functional units are placed at Manhattan distance from one another with a target aspect ratio determined by the system floorplan. Although basic, Ctherm's routines can be extended with techniques such as [15] for fast thermal-aware floorplanning of tiles, and [16] for minimizing the wirelength

```
−−−−−−−−−−−−−−−Caches−−−−−−−−−−−−−−−
position ← initAnchorPosition
for currentUnit in list(cacheFunctionalUnits) do
    (dimensions, pitch, count) ← ALEmodel
    (maxRows, maxColumns) ← ALEmodel
    for rows in 0 to maxRows do
        for cols in 0 to maxColumns do
            PLACEUNIT(id, position, dimension)
            UPDATE(position)
−−−−−−−−−−−Interconnect Routers−−−−−−−−−−
(dimensions, ports, TSV_count) ← ALEmodel
position ← initAnchorPosition
centerPoint ← CALC(initAnchorPosition,dimensions)
position ← centerPoint
for currentUnit in list(interconnectFunctionalUnits) do
    if currentUnit == Crossbar then
        PLACEUNIT(id, position, dimension)
    else if currentUnit == I/O Port then
        for currentPort in list(N, S, E, W, L, Up, Down) do
            newPosition ← CALC(position,dimensions)
            position ← newPosition
            PLACEUNIT(id, position, dimension)
    else
        Perform TSV Topology Exploration
        PLACETSV(TSV_count, generated_topology)
−−−−−−−−−Processing Elements−−−−−−−−−
(dimensions) ← ALEmodel
position ← initAnchorPosition
for currentUnit in list(processorFunctionalUnits) do
    PLACEUNIT(id, position, dimension)
    nextPosition ← CALCMANHATTAN(position, aspectRatio)
    position ← nextPosition
```

Fig. 5. Routines for generation of fine-grained cache memory, interconnect router and processing element floorplans.

between functional units. Furthermore, support for components such as programmable accelerators can be added through the inclusion of additional planning routines. The floorplanner computes the overall die size, and generates a die descriptor containing a physical description of the die and its material properties, in addition to the detailed system floorplan. For 3D stacked SoCs, the floorplanner is executed iteratively till floorplans for each die have been generated, and the stack descriptor describes each die of the stack together with a path to its corresponding floorplan. This descriptor also includes the inter-tier thermal conductivity corresponding to the chosen TSV count and topology.

B. Thermal-Functional Co-simulation Platform

The second stage of the Ctherm framework performs the thermal-functional evaluation of the SoC using the generated physical model, and the input SystemC top-level file configured with the system specifications. The co-simulator consists of a cycle-accurate simulation engine integrated with an embedded thermal simulator. The simulation engine instantiates components from the *SoCLiB* IP library [17] which consists of an extensive set of SystemC behavioural models for processor cores, interconnects, caches, memories, controllers and accelerators. Thermal simulation of modeled platforms is enabled by an adapted version of the *3D-ICE* thermal simulation engine [18], embedded within the co-simulation platform core.

1.) Power Mapper: A thermal model for the system is generated based on the die descriptor and floorplan generated in the previous stage of the framework. The die is discretized into a grid of thermal cells, with cell size determining the resolution of the resulting thermal maps, and also complexity of the thermal model. The thermal model generation step is performed only once per simulation run, and usually completes in under a minute for cell sizes of 50μm. Logging of thermal maps on the other hand poses a significant overhead that is dependent on thermal cell size. However, since temperatures on die do not change at the one-cycle time scale, a logging interval of 50μs provides sufficient resolution for visualization of hotspots in most simulations, with an acceptable overhead.

SystemC components of the SoCLiB IP library are augmented with an activity tracking function that logs the operations performed by their constituent functional units on a cycle-accurate basis. All activity frames are evaluated at the start of every thermal simulation time step, and are converted into a detailed power map using the corresponding ALE model for each component. Since the model contains internal organization details of components, an exact list of units activated and their corresponding power dissipation can be computed by the power mapper.

The thermal simulation is triggered once activity data has been collected from all components, at a rate determined by the thermal time step. Time steps larger than the execution step requires aggregation or averaging of power maps until insertion, which can result in false hotspots and false blurring of hotspots respectively.

2.) Checkpointed Thermal Simulation: Thermal simulations are normally carried out for the same duration of time as the functional simulation. However, in some cases, the thermal behaviour that we want to characterize can occur much later in the simulation. In order to observe this behaviour in isolation, Ctherm supports the discrete starting and stopping of the thermal simulation engine at any time during the execution. This is achieved through the *Thermal Simhelper* component which acts as an interface between the virtual and co-simulation platforms, enabling control of the thermal simulation both from the behavioural model of components, as well as from software executing on the virtual platform. With thermal simulation disabled, functional simulation speed increases by a minimum of 30%.

A drawback of this approach is that it results in the loss of thermal continuity until the point thermal simulation is enabled. Therefore, fast-forwarding from the start of functional simulation results in the thermal simulation starting with a die at initial temperature (ambient). To overcome this, we integrated the ability to save thermal checkpoints to disk using the Thermal Simhelper. Furthermore, we adapted the 3D-ICE core to initialize the system's thermal model using user-specified checkpoints. Consequently, Ctherm allows the saving of thermal checkpoints to disk during simulation, and the initialization of the system's thermal state with any user specified checkpoint during the simulation. In order to do this, thermal simulation is first performed for the time interval that will be fast-forwarded, and a thermal checkpoint is created at the end, as illustrated in Figure 6. Subsequent thermal simulations can begin directly at the point of interest, after initializing the thermal state with the saved checkpoint. Once initialized, the system appears as if thermal simulation has been running since the beginning. Checkpointed simulations can drastically cut time spent in simulating specific temporal effects, and testing systems post optimization.

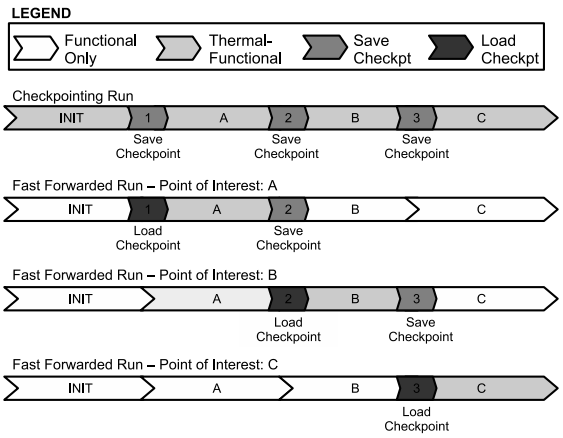

Fig. 6. Illustration of checkpointing run generating thermal checkpoints, followed by fast forwarded runs using saved thermal checkpoints.

In addition to thermal checkpointing, Simhelper also allows the temperature map of the die to be saved and read at any point during the co-simulation. These maps can be sampled at discrete locations to emulate temperature sensor readouts.

III. Evaluation

The effectiveness of Ctherm is illustrated using real design cases, each performed on an Intel PentiumD 3.0GHz machine with 4GB memory. *CACTI* [12], *WATTCH* [11] and *ORION* [13] are used as ALE estimators for caches, processors and the system interconnect respectively. All design cases consider the $90nm$ technology node, a 200MHz clock frequency and a $340K$ $(67°C)$ critical temperature. The results of the evaluation are presented in two parts, starting with the accuracy of thermal characterization and simulation speed. This is followed by the four design cases.

A. Validation, Accuracy and Simulation Speed

Since Ctherm implements changes to the 3D-ICE core in order to enable thermal checkpointing, the effect of these modifications was validated against an unmodified version. The two versions were found to produce matching thermal profiles, and the implemented modifications were found to induce no errors or variations in the thermal simulation results. The modified version therefore has the same accuracy and correctness as the standard 3D-ICE core. To verify the power mapper, the total power inserted per operation during the simulation was validated against the values reported by the ALE estimators, and the locations of these insertions were compared against manually computed locations and verified to be correct across different system configurations.

Ctherm improves thermal simulation accuracy by using generated fine-grained internal floorplans for components. This enables the accurate distribution of dissipated power across the constituent functional units of components. To illustrate the advantage such fine-grained modelling provides, we examine

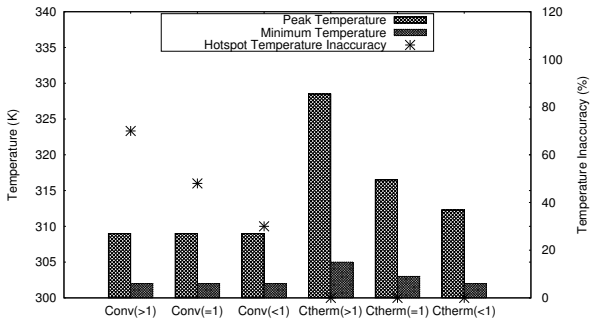

Fig. 7. Comparison of thermal simulation with abstract floorplans (*Conv*) and thermal simulation using Ctherm's fine grained floorplans (*Ctherm*) for a cache memory with varying wordline sharing factor (N_{spd}). Three cases: $N_{spd} > 1$, $N_{spd} = 1$, $N_{spd} < 1$.

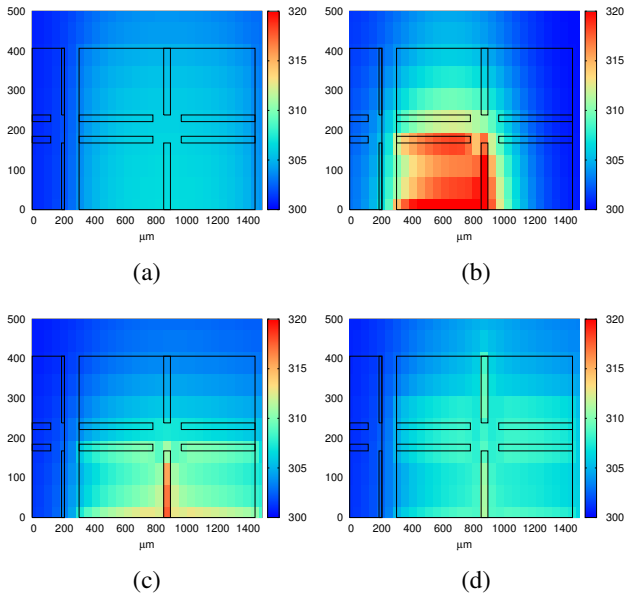

Fig. 8. Heatmaps for 32KB data cache using (a) conventional thermal-simulation with abstracted cache internals, and Ctherm with fine-grained cache floorplans for (b) $N_{spd} > 1$, (c) $N_{spd} = 1$, (d) $N_{spd} < 1$. Temperatures are measured in *Kelvin (K)*.

the influence of varying the wordline sharing factor (N_{spd}) on the temperature profile of a 32KB data cache following $1E5$ sustained accesses to a single cacheline. This evaluation follows the illustration previously shown in Figure 3.

The three cache organizations ($N_{spd} > 1$, $N_{spd} = 1$, and $N_{spd} < 1$) are first evaluated using the conventional thermal simulation approach (*Conv*), i.e. abstracting component internals. This abstraction results in the distribution of dissipated power across the complete area of the cache, and thus the peak and minimum temperatures for the three are identical as observed in Figure 7. The organizations are subsequently evaluated using Ctherm's fine-grained floorplans which account for the N_{spd} parameter. The results of this evaluation reveal remarkable differences in the peak temperature for each case, indicating the presence of a hotspot especially in the case of $N_{spd} > 1$. The difference in temperature profiles as a result of varying distribution of dissipated power are clearly visible in Figure 8b-d. The temperature map obtained with abstracted component internals in Figure 8a on the other hand incorrectly reports peak temperatures upto 70% lower than those obtained with fine-grained simulation. This highlights the importance of fine-grained floorplans during tDSE and thermal-aware system design.

TABLE I. SIMULATION SPEED ACROSS DESIGN CASES LISTED IN SECTION III-B

Case	*Tcell Size* (μm)	*Die Size* ($\mu m \times \mu m$)	*Sim. Speed* (*cycles/min*)
1 (FP_A,FP_B)	100	1400×1200	200K
1(FP_C,FP_D)	100	700×1200 ($\times 2$)	200K
2	100	1400×1200	400K
3	50	1350×850	215K
4	100	700×600	600K

The speed of the Ctherm framework largely depends on the complexity of the system being evaluated, granularity of thermal simulation and die size. In order to provide an idea of the average simulation speed in realistic design scenarios, we report the runtime Ctherm incurred in performing the four design cases listed in Section III-B. The simulation speeds are reported in Table I.

B. Design Cases:

In this subsection, we illustrate the applicability of Ctherm using four specific design cases:

1.) 2D/3D Floorplan Selection: On account of severe space constraints in portable devices, embedded MPSoCs cannot afford to have extravagant heatsinks, and it is therefore important for them to be designed for operation within an extremely narrow thermal envelope. This design case involves the evaluation of floorplan options for a multiprocessor array with four *processing elements (PE)*. Candidate floorplans for the array are illustrated in Figure 9(a). Each floorplan depicts four tiles containing a simple RISC PE, private *4KB 2-way* instruction and data caches, a $64b$ 5-port network-on-chip router connecting to neighbouring tiles and a temperature sensor at the location marked as $+$. A *Dynamic Thermal Management (DTM)* scheme is included in PE tiles to disable switching activity as soon as the critical temperature is breached, with a reactivation temperature margin of $2K$. The $dijkstra$ shortest-path benchmark from the $MiBench$ suite [19] is used as a test workload for the virtual platform. The thermal and functional performance of the system for each floorplan is reported in Table II.

Average Off Time indicates the fraction of the simulation time for which PEs remained disabled due to a DTM action, thus indicating the thermal efficiency of floorplans. FP_B's spreading out of PEs is seen to result in decreased average off time, causing a decrease in the number of cycles taken to execute the workload, i.e. *cycles per instruction (CPI)*. Figure 9(b) illustrates the thermal maps for each floorplan.

TABLE II. THERMAL-AWARE PERFORMANCE ESTIMATES FOR FLOORPLAN EXPLORATION. * MARKS THERMAL RUNAWAY.

	FP_A	*FP_B*	*FP_C*	*FP_D*
Cumulative CPI	1.07	1.05	1.33	1.24
Data Refs/cycle	0.265	0.270	0.215	0.229
Avg. Off Time	55%	45%	100%*	100%*

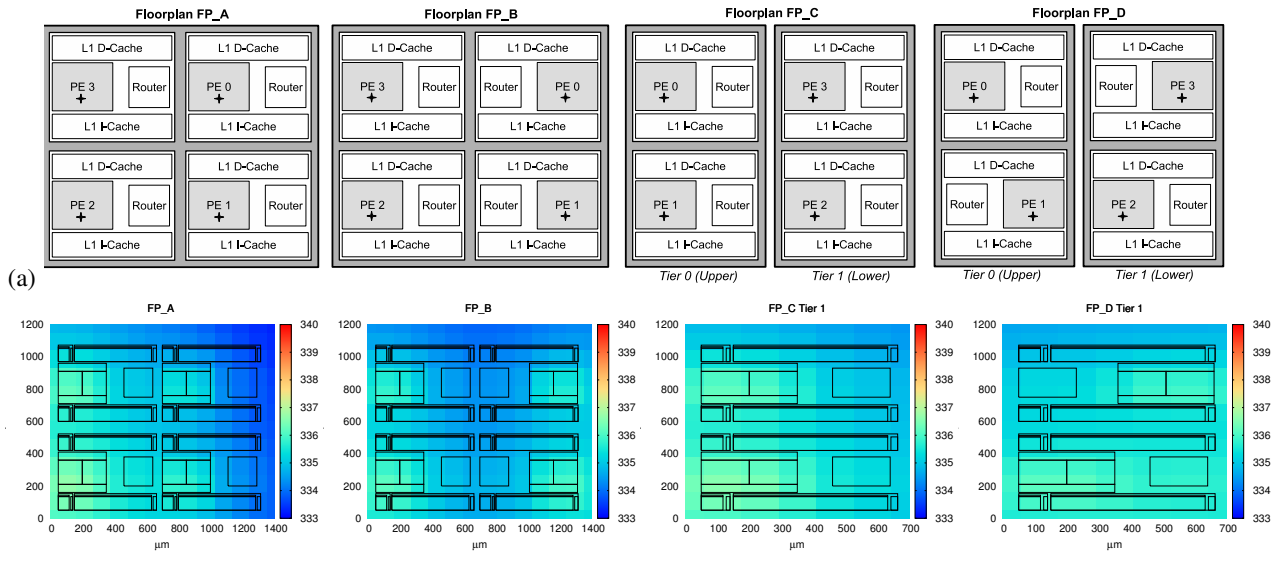

Fig. 9. (a) Floorplan options for the four-PE multiprocessor array. Temperature sensor locations are marked as $+$. (b) Thermal maps sampled during execution of the *MiBench-dijkstra* workload. Note: Tier 0 is located close to the heatsink/connection to ambient, temperatures are measured in *Kelvin (K)*.

These were automatically generated by Ctherm together with the floorplan overlay. The 3D options FP_C and FP_D split the four processors over two dies, thus halving the overall area footprint for the system, without changing the total power dissipation. Since the power dissipated by PEs is conducted to the ambience through the surface of the die, this reduction results in decreased cooling efficiency. In fact, the cooling efficiency of the 3D configuration is constrained to such an extent that even after disabling all switching activity in the system, temperatures continue to rise due to the leakage power dissipation. Floorplans FP_C and FP_D thus encounter a thermal runaway. This result indicates that in order to use the 3D design points, either cooling efficiency must be improved, or leakage control/power gating mechanisms must be integrated into the architecture to limit leakage power dissipation especially in lower tiers of the stack.

2.) Thermal-aware Architecture Exploration: An architectural scheme is described in [20], that uses a small fully associative assist cache to decrease the average latency and energy for data cache accesses. We evaluate the thermal impact of such a cache assist on the performance of uniprocessor SoC consisting of a single PE, a *64KB 4-way* data cache and an *8KB 2-way* instruction cache. A multi-dimensional array implementation of the *first sum (kernel11)* workload from the *Livermore Loop Kernels* benchmark [21] is executed on the platform, with an array size of 168×168. The assist scheme is observed to reduce the average latency of memory accesses, and consequently CPI by upto 24% compared to a conventional data cache. While this improves performance, it results in execution proceeding at a faster rate, leading to a quicker ramp up of temperatures as seen in Figure 10. However, since CPI is reduced, execution performance completes approximately 1 million cycles earlier, leading to a peak temperature that is $0.9K$ lower than the conventional data cache.

3.) Temperature Sensor Placement: The ability to monitor die temperature is a prerequisite for performing runtime

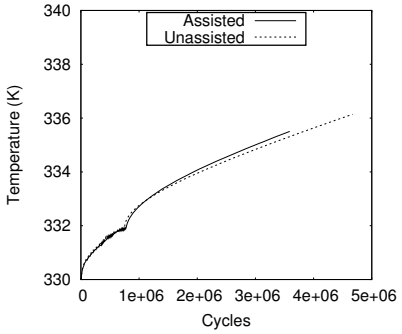

Fig. 10. Comparison of temperature profiles obtained using a conventional data cache (*Unassisted*), and a data cache with an 8-entry assist (*Assisted*).

thermal management. Temperature sensors usually integrate an analog-to-digital converter or a digital-to-analog converter [22] for accurate digital readouts. Their size however limits the number, as well as the locations at which they can be placed. Sensors located far from hotspots exhibit delayed responses and inaccurately measure hotspot temperature, as we illustrate in this design case. It is therefore prudent to evaluate placement options for sensors by using thermal-functional co-simulation of the system. In the event that a sensor cannot be placed close to a hotspot, results of the co-simulation yield information on how to calibrate the DTM's temperature margins.

Figure 11(a) shows a $1350\mu\text{m} \times 850\mu\text{m}$ die with a single PE and its caches. The ideal temperature sensor location is marked with a $+$ symbol, while candidate locations for sensor placement are indicated with labels A through F. Locations shown inside the caches are situated in wiring tracks with unused active regions. Figure 11(b) plots the rise in PE temperature as a function of time, and also the perceived temperature rise at each sensor location. Due to their distance from the PE, sensors at C and F exhibit a delayed response,

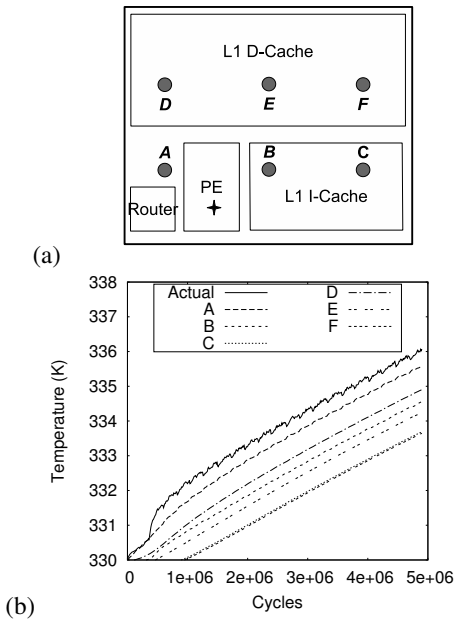

Fig. 11. (a) Floorplan indicating candidate locations for temperature sensors. (b) Tracking from each sensor location.

TABLE III. COMPARISON OF CONVENTIONAL AND CHECKPOINTED THERMAL SIMULATION RUNTIME

	ITERATIONS		RUNTIME (SECONDS)	
	Conv	*Checkpointed*	*Conv*	*Checkpointed*
Initialization	20	1	-	159
Kernel A	10	10	2060	760
Kernel B	10	10	4640	3340
Total Runtime (seconds)			*6700*	*4259*
Improvement over Conventional Approach				**36**%

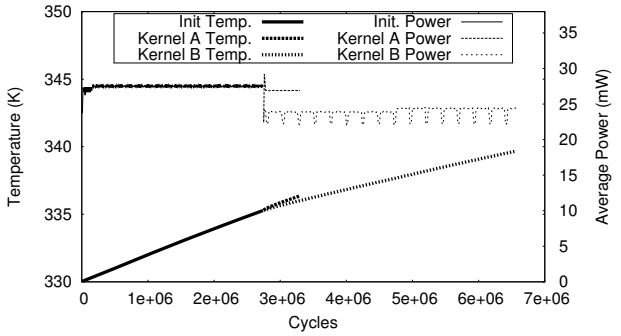

Fig. 12. Temperature and power profiles for initialization, kernel A and kernel B of the extended *first sum* workload

and report temperature readings $2K$ lower than the actual. A DTM using sensors at these locations must account for this inaccuracy in its temperature margins in order to effectively control system temperature.

4.) Thermal Impact of Software Workloads: The thermal behaviour of processors in SoCs is largely determined by the workloads that execute on them. Minor changes to the software algorithm can have a drastic impact on both execution performance as well as system power and thermal profiles. To illustrate this, we extend the *first sum* workload of design case 2 with an additional kernel option (B). The original kernel (A) performs the first sum computation in a column-first manner, while the new kernel (B) follows a row-first approach. The kernels are precluded by an initialization of the multidimensional arrays, which executes for approximately $2.75M$ cycles, following which the computation begins. In order to emulate the iterative nature of optimizations and thermal evaluations for workloads, each kernel's evaluation was repeated 10 times. Two sets of runs were performed, one using the conventional continuous thermal simulation, and another using Ctherm's checkpointed simulation.

The conventional approach without checkpointing requires the simulation to be restarted from the beginning of execution following each optimization to the kernel. This means that in addition to the kernel under test, all other sections of the program must also be re-simulated. Thus the initialization section is simulated a total of 20 times, 10 times for each kernel. Note that since each simulation in the conventional approach causes the entire program to be executed, the runtime of the initialization section is included within that of the kernels. With the checkpointed approach, on the other hand, the initialization is simulated only once and its resulting thermal map saved as a checkpoint. This checkpoint is subsequently used as a starting point for each kernel's thermal simulation, thereby allowing the initialization to be fast-forwarded. Total runtime is consequently decreased by 36% as observed in Table III The temperature and power profiles for the initialization and the two kernels are reported in Figure 12. The row-first approach of kernel B results in a high miss-rate observable from the repeated fluctuations in its power trace. This consequently increases runtime, resulting in a higher leakage energy consumption, and a peak temperature over 4K higher than that of kernel A.

IV. CONCLUSIONS

This paper presented Ctherm, an integrated framework for thermal-functional co-simulation of systems-on-chip. Ctherm enables the characterization of internal component thermal behaviour by using fine-grained physical models for components, automatically generated from input system specifications. Ctherm's fine-grained modelling improves accuracy of hotspot temperature resolution by upto 70% as compared to the conventional approach that abstracts component internals. Furthermore, by introducing thermal checkpointing, the Ctherm framework enables discrete thermal simulations which reduce the runtime of post-optimization evaluation runs by upto 36% over the conventional continuous simulation approach. The efficacy and applicability of the framework were illustrated with four realistic design cases.

ACKNOWLEDGMENT

This research was supported in part by the CATRENE programme under the Computing Fabric for High Performance Applications (COBRA) project CA104. The authors would like to thank Michel Berkelaar for his inputs on floorplan generation for cache memories, and Jurrien de Klerk for his assistance with the instruction energy estimation.

REFERENCES

[1] W. Heirman, S. Sarkar, T. E. Carlson, I. Hur, and L. Eeckhout, "Power-aware multi-core simulation for early design stage hardware/software co-optimization," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, 2012, pp. 3–12.

[2] K. Skadron et. al, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Transactions on Architectural Code Optimizations*, vol. 1, no. 1, pp. 94–125, 2004.

[3] G. Paci et. al, "Exploring "temperature-aware" design in low-power mpsocs," in *Proceedings of the Design, Automation and Test in Europe Conference Exhibition*, 2006, pp. 838–843.

[4] D. Atienza et. al, "Hw-sw emulation framework for temperature-aware design in mpsocs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 12, no. 3, pp. 26:1–26:26, May 2008.

[5] A. Bartolini et. al, "A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores," in *Proceedings of the Great Lakes Symposium on VLSI*, 2010, pp. 311–316.

[6] T. Bouhadiba et. al, "Co-simulation of functional systemc tlm models with power/thermal solvers," in *Proceedings of the International Symposium on Parallel and Distributed Processing*, 2013, pp. 2176–2181.

[7] A. Varma, *PhD Dissertation: High-speed Performance, Power and Thermal Co-simulation for SoC design*. University of Maryland, 2007.

[8] S. Priyadarshi et. al, "Thermal pathfinding for 3-d ics," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 4, no. 7, pp. 1159–1168, 2014.

[9] D. Milojevic, T. Carlson, K. Croes, R. Radojcic, D. Ragett, D. Seynhaeve, F. Angiolini, G. Van der Plas, and P. Marchal, "Automated pathfinding tool chain for 3d-stacked integrated circuits: Practical case study," in *Proceedings of the IEEE International Conference on 3D System Integration*, 2009, pp. 1–6.

[10] J. Cong et. al, "An automated design flow for 3d microarchitecture evaluation," in *Proceedings of the Asia and South Pacific Design Automation Conference*, 2006, pp. 384–389.

[11] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proceedings of the International Symposium on Computer Architecture*, 2000, pp. 83–94.

[12] N. Muralimanohar et. al, "Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0," in *Proceedings of the International Symposium on Microarchitecture*, 2007, pp. 3–14.

[13] 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 *Proceedings of the Design, Automation, Test in Europe Conference Exhibition*, 2009, pp. 423–428.

[14] S. S. Kumar, A. Aggarwal, R. Jagtap, A. Zjajo, and R. van Leuken, "System level methodology for interconnect aware and temperature constrained power management of 3-d mp-socs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 7, pp. 1606–1619, July 2014.

[15] D. Cuesta, J. Risco-Martin, J. Ayala, and D. Atienza, "3d thermal-aware floorplanner for many-core single-chip systems," in *Proceedings of the Latin American Test Workshop (LATW)*, 2011, pp. 1–6.

[16] D. Cuesta, J. Risco-Martin, J. Ayala, and J. Hidalgo, "3d thermal-aware floorplanner using a moea approximation," *Integration, the VLSI Journal*, vol. 46, no. 1, pp. 10–21, 2013.

[17] SoCLiB-Project, "Soclib: an open platform for virtual prototyping of multi-processors system on chip." [Online]. Available: http://www.soclib.fr

[18] A. Sridhar et. al, "3d-ice: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling," in *Proceedings of the International Conference on Computer-Aided Design*, 2010, pp. 463–470.

[19] M. R. Guthaus et al., "Mibench: A free, commercially representative embedded benchmark suite," in *Proceedings of the International Workshop on Workload Characterization*, 2001, pp. 3–14.

[20] S. S. Kumar and R. van Leuken, "Improving data cache performance using persistence selective caching," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1945–1948.

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

[22] A. Zjajo, N. van der Meijs, and R. van Leuken, "A 11 μw 0°c-160°c temperature sensor in 90 nm cmos for adaptive thermal monitoring of vlsi circuits," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2012, pp. 2007–2010.