Run-Time Adaptable On-Chip Thermal Triggers

Pratyush Kumar David Atienza

Embedded Systems Laboratory (ESL), EPFL
1015 Lausanne, Switzerland
e-mail: {pratyush.kumar, david.atienza}@epfl.ch

Abstract— With ever-increasing power densities, Dynamic Thermal Management (DTM) techniques have become mainstream in today's systems. An important component of such techniques is the thermal trigger. It has been shown that predictive thermal triggers can outperform reactive ones [4]. In this paper, we present a novel trade-off space of predictive thermal triggers, and compare different approaches proposed in the literature. We argue that run-time adaptability is a crucial parameter of interest. We present a run-time adaptable thermal simulator compatible with arbitrary sensor configuration based on the Neural Network (NN) simulator presented in [14]. We present experimental results on Niagara UltraSPARC T1 chip with real-life benchmark applications. Our results quantitatively establish the effectiveness of the proposed simulator for reducing (by up to 90%), the otherwise unacceptably high errors, that can arise due to expected leakage current variation and design-time thermal modeling errors.

I. Introduction

Power densities in today's highly integrated systems like Multi-Processor Systems-on-Chip (MPSoCs) continue to increase primarily due to three fundamental factors: a) continuing, albeit slower, decrease in feature sizes, b) increasing functional complexity of such systems, and c) slower rate of improvement in hardware cooling solutions [24]. Higher power densities directly lead to higher operating temperatures, which can result in not only long-term reliability issues but can affect performance and/or correctness.

Architectural level control techniques, broadly referred to as Dynamic Thermal Management (DTM) [2, 8], have emerged as a necessary requirement in today's MPSoCs. In essence, DTM techniques aim to smartly manage the computation load on on-chip resources so as to avoid thermal hazards. According to the classification suggested in [2], a DTM technique comprises of three components: a thermal trigger, a response, and a controlling policy. The thermal trigger indicates the breach of a certain pre-defined thermal threshold. Different choices of thermal triggers include temperature readings from sensors, architectural-level performance counters and compile-time analysis. The response is the actuator by which the system attempts to reduce the temperatures on-chip. Examples of such responses include dynamic voltage/frequency scaling (DVFS) [16], clock-gating [27], task migration [19], liquid coolant control [5] and architecture-specific throttling such as speculation control [10] and I-cache toggling [2]. The control policies decide when to turn on and off the different responses. This modular view of a DTM technique allows different trigger mechanisms to be combined with different choices of responses. For instance, DVFS can be used independent of whether the thermal trigger is based on on-chip thermal sensor measurements or computed from architectural-level performance counters. The focus of this work is to study the space of different trigger mechanisms, independent of the response component.

As analyzed in [2], *reactive* DTM techniques have a finite initiation and response delay, which is incurred between a thermal trigger and the invocation of the corresponding response. To avoid thermal hazards in the face of such delays, the thermal thresholds have to conservatively reduced, thereby impacting the guaranteed performance of the system. In [25], it has been shown that *predictive* triggering mechanisms can greatly outperform naive reactive triggering mechanisms. In [4], it has been shown that with proactive speed control, the real-time guarantees of a system can be improved.

Directly reading thermal sensor values are naturally reactive. We categorize the other thermal triggers proposed in the literature into five classes as detailed in Section II. Accuracy of such predictive triggers is important, since any errors would translate to conservatively approximated performance guarantees of the system. Further, since the prediction is performed on-chip, it is essential to be computationally efficient and use minimal resources. A natural trade-off, thus, exists between accuracy and computational efficiency. However, another crucial parameter of interest in run-time adaptability. As we shall discuss in Section II, including run-time adaptability greatly skews the trade-off space of different trigger mechanisms. Run-time adaptation in on-chip thermal triggers is required for the following two important reasons.

- Due to process variations, the leakage current can significantly differ from design-time expected values [1]. A higher leakage current translates to higher leakage power and thus larger temperatures. As we illustrate in the experimental results of Section IV, the corresponding errors in the computed temperatures for a real MPSoC can be unacceptably high.

- The model of heat dissipation on the chip depends on the thermal contact between the chip and the ambient: a parameter that is difficult to estimate. This can lead to errors in the thermal model of the chip. We quantify the deviation in computed temperatures in the presence of such errors in Section IV.

In this paper, we propose a Neural-Network (NN) based on-chip thermal simulator as an adaptable predictive thermal trigger. This is an extension of the simulator we proposed in [14]. In this paper, we make two crucial extensions, to the proposed simulator of [14]. Firstly, it is assumed in [14], that a fine and regular grid of sensors is available on-chip. In reality however, only a few thermal sensors, in a possibly irregular arrangement, are available on-chip. We adapt the simulator to work with any given thermal sensor layout. Secondly, an on-chip refinement method to adapt the NN simulator has not been detailed in [14]. We propose a low-complexity back-propagation-based method to refine the NN with minimal additional hardware. With such a refinement the NN can potentially correct any design-time model errors.

The contributions of the paper are on three fronts:

- formalizing the trade-off space of predictive thermal triggers and placing on it different state-of-the-art techniques proposed in the literature,
- extending the NN-based simulator proposed in [14] to be applicable as an adaptable predictive thermal trigger for a real system, and
- quantitatively establishing the need for run-time adaptability in on-chip thermal triggers. We demonstrate that errors in temperature can be reduced by up to 90%.

The rest of the paper is organized as follows. In Section II, we present the aforementioned trade-off space of predictive thermal triggers and discuss the range of related work in the field. In Section III, we propose the said two extensions to the NN simulator presented in [14]. In Section IV, we provide quantitative results on a real MPSoC platform.

II. Trade-off space of predictive thermal triggers

We first classify the different predictive thermal triggers proposed in literature into five categories: (a) design-time analytical models, (b) software-based thermal simulators, (c) triggers based on model-predictive control, (d) workload-predictive triggers, and (e) hardware-based thermal simulators. We discuss each of these categories with respect to the three criteria: computational efficiency, accuracy guarantees and run-time adaptability.

Design-time analytical models are used in triggers where future events are predicted based on design-time models of the thermal system and/or application. Such triggers are used in convex-optimization-based frequency assignment [21] and control-theoretic DTM [23]. These methods compute and optimize in design-time and export only the required parameters for on-chip computation. Thus, they are computationally efficient. They are also accurate as the design-models are based on the exact equations of heat transfer. However, these methods are severely limited by having no run-time adaptability at all.

Model predictive control techniques have been applied to DTM techniques in [9]. The predictive nature of the trigger comes from the derived analytical model of the thermal system. However, unlike in [23], the model is refined based on the

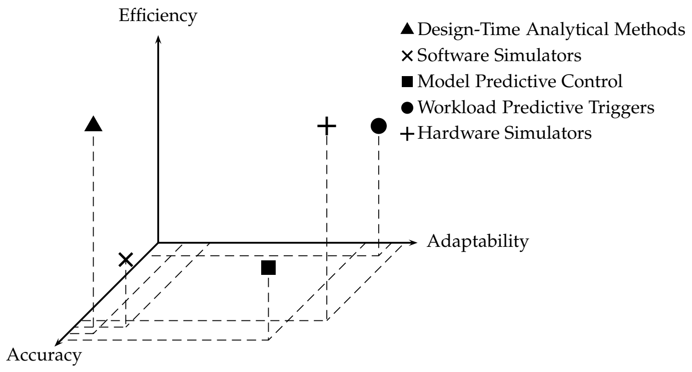
Efficiency
Adaptability
Accuracy
▲ Design-Time Analytical Methods
× Software Simulators
■ Model Predictive Control
● Workload Predictive Triggers
+ Hardware Simulators

Fig. 1. Qualitative representation of the trade-off space of predictive thermal triggers. In each dimension, higher values are better.

readings of on-chip thermal sensors. Thus, the method allows for accurate simulation with run-time adaptability. However, the on-chip deployment of such a technique is not discussed in [9]. The complicated data structures and computation involved in the method would potentially require a software only deployment on-chip with large resource usage.

Software-based simulators have been the focus of several studies where algorithms and data-structures are studied to optimize the thermal simulation of a chip. Examples of these include, amongst several others, HotSpot [11], 3D-ADI [26] and ISAC [28]. These methods allow for very accurate simulation of the system. However, all of these methods are designed for use in design-time thermal-aware optimizations. Implementing them on-chip would incur either large resource overhead or large computational delays. Furthermore, no obvious approaches exist to adapt these algorithms during run-time.

Workload-predictive triggers aim to predict the quantum of the workload on the system in the future and thereby compute the temperatures on-chip. In [6, 13], stochastic methods are used to approximate the temperatures of the system assuming a stationary workload model. In [25], temperatures in the future are computed for frame-based multimedia applications, where the workload model is well-defined. Such methods are computationally efficient and allow for run-time adaptability. However, since these methods are not based on the exact equations of heat flow, their key limitation is the lack of generic accuracy guarantees. Each application has to be tested to confirm acceptable accuracy.

To the best of our knowledge, the only hardware-based thermal simulator studied is the Neural Network (NN) based thermal simulator presented in [14]. The key advantages of this method are small computational delays (in the order of a few gate delays), low resource usage (of about 0.1% of the total chip transistor count), and low error margins (of 1-2K). In the later sections of this work, we extend this simulator to adapt during run-time and work for arbitrary sensor layout configurations.

In summary, design-time analytical models and software-simulators-based thermal triggers do not provide any adaptability on-chip. Adaptability can be obtained using sophisticated techniques like model predictive control at the cost of computational efficiency and resource overhead, or using stochastic workload-based techniques at the cost of reduced robustness in accuracy guarantees. NN-based simulator competitively outperforms all other methods at least in one key parameter. We qualitatively plot the different classes of predictive thermal triggers in Fig. 1.

III. Runtime adaptable thermal simulator

A. Neural-network-based thermal simulator

The NN simulator presented in [14] is based on two principles: a) the thermal system is Linear Time Invariant (LTI), and b) this linear dependence can be learnt using a NN. The thermal system as represented by the compact model is characterized by the following equation

$$GT(t) + C\dot{T}(t) = P(t) \tag{1}$$

where G and C are the compact model parameters and T and P are vectors that denote, respectively, the temperature and power at a given time in a grid of points, as used in the compact model. Using the LTI property, the above can be reduced to the following equation

$$T(t_{n+1}) = AT(t_n) + BP(t_n) \tag{2}$$

where A and B are matrices which are derived from the compact model parameters. The above equation can be *solved* by training a linear NN, which can be expressed generally as

$$y_j = \sum_i w_{ij} x_i \tag{3}$$

where y and x are vectors that denote the outputs and inputs, respectively, and w denotes the weight terms, which would learn the A and B matrices of (2). For a given system, the weight terms can be learnt by training using data obtained from real chip measurements or obtained from simulation results on a tool like Hotspot. Results in [14] show that such a NN, once trained offline, can be use to simulate the system in time-hops of about 0.5 s, with small error margins. A crucial advantage of such an approach is that a NN can be inexpensively fabricated on-chip using an array of weighted current mirrors. Each of the weight terms, w_{ij} may be represented by say b binary bits. We showed in [14] that with a negligibly small fraction of transistor count of the entire chip, a NN can be designed that can simulate the temperature of the entire chip.

B. NN simulator for arbitrary thermal sensor layout

In [14] the NN simulator that was applied to a uniformly distributed grid of points on a chip. This uniform distribution was based on the granularity of the corresponding compact model. For accurate results, a fine compact model is used. However, a fine model implies that temperature data of a large number of points need to be fed into the NN as inputs. This is a reasonable assumption for simulation-based off-chip training of the NN. However, for on-chip training, which is the goal of this work, we cannot expect such detailed thermal data. Only a few thermal sensors are fabricated on-chip: a decision based on the non-negligible area and cost overheads of thermal sensors. Hence, it is necessary to adapt the NN simulator to work with different sensor layout configurations, and to quantify the accuracy of the simulation for such configurations.

Research on placement of sensors on-chip is not yet conclusive. One approach is to place the sensors so that they cover the hot regions of the chip [15]. Another approach is to perform clustering based optimizations to place sensors which attracts them to the thermal gradients [20]. In [18] the authors also suggest a regular layout of sensors with an interpolation operation to reduce errors. We study three representative sensor layout configurations: a) `Reg`, b) `HS`, and c) `Rand`. In `Reg`, we place the sensors in a regular grid around the chip. This leads to maximum area coverage and minimum reliance on off-chip data. In `HS`, we place the sensors in the hottest regions of the chip, which are identified based on simulations of common benchmarks. Finally, we also consider `Rand` layouts where several randomized layouts are considered, and all results are averaged across the different layouts. In all cases, to facilitate comparison, we consider a constant number of sensors.

Adapting the NN simulator to work with a given thermal sensor layout is equivalent to a Model Order Reduction (MOR) of the set of differential equations represented in (1) to a reduced set of equations involving only the temperatures of the points covered by the sensors. This requires finding out effective thermal parameters - conductance and capacitance matrices - networking these sensor points. This can be performed by analyzing the RC network, and performing standard projection based MOR as studied for RLC interconnect networks in [17]. The reduced set of equations would define effective matrices A_r and B_r such that

$$T_s(t_{n+1}) = A_r T_s(t_n) + B_r P_s(t_n) \tag{4}$$

where T_s and P_s are vectors that denote the temperature and power consumption, respectively, of the points covered by the thermal sensors.

Note that the MOR step retains (a) the linear nature of (4), and (b) the time-invariant nature of the system. Thus, the MOR operation is also LTI. Composition of two LTI operations is also LTI. Hence, we can design an *augmented* NN, that performs both the MOR and the simulation of temperature, inherently. Such a NN would take as inputs the current temperature and power consumption of points covered by sensors and provide their future temperatures as outputs. Thus, by simulating a model in a simulator like HotSpot or by using real-chip data, we can train the NN for temperatures of only those points where sensors are fabricated, and thereby learn the weight matrices A_r and B_r.

C. Run-time adaptation in NN simulator

To adapt the NN on-chip in the face of variations discussed earlier, we must refine the underlying NN model during runtime. This would require that the weight terms, w_{ij} of the NN that model the A_r and B_r matrices, be programmable. In other words, the b bits used to represent each weight term, should be stored digitally and be modifiable. We can then use back-propagation learning algorithms [3] to adapt the weights as follows:

$$\Delta w_{ij} \leftarrow \Delta w_{ij} + (y_i^* - y_i) y_j \tag{5}$$

$$w_{ij} \leftarrow w_{ij} + \mu \Delta w_{ij} \tag{6}$$

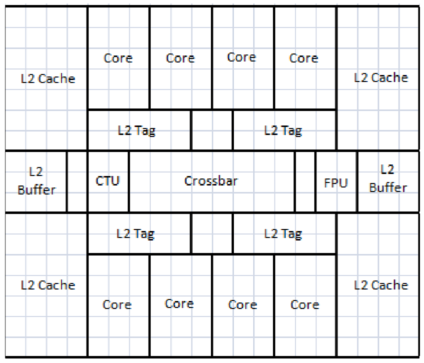

Fig. 2. Floorplan of the Niagara chip with 340 cells for compact modeling

where y_i^* is the correct value of the i^{th} output and μ is the learning rate. We adopt the batch learning rule [3], wherein (5) is executed several times before updating the weight terms as shown in (6). The Δw terms are then reset to 0.

It is important to distinguish between off-chip and on-chip training of the NN. Note that the least normalized unit of the weight terms is given by 2^{-b}, where b is the number of bits used in the representation of the weight terms. In off-chip training, this quantization needs to be imposed only during the final translation of the computed weight terms to the required b bits. However, during on-chip training, each of the updates of (6), would enforce a quantization. With this quantization, the accuracy of the simulator can potentially reduce. We quantify the loss in accuracy in Section B.

It is important to note that the only additional hardware requirement of this extension is that b bits be stored for each weight term. The binary weighted current mirror based multipliers as proposed in [14] remain the same. For instance, for a chip with 12 sensors fabricated and with $b = 8$, the additional storage requirement is 264 bytes. This is an acceptably small overhead for the crucial property of run-time adaptability.

IV. Experimental results

A. Target system and benchmark applications

Our case study, as in [14], is based on the 8-core UltraSPARC T1 (Niagara-1) architecture from Sun Microsystems [12]. This MPSoC has been manufactured in 90-nm process technology. In each core, four threads share an integer pipeline. Every two cores share an L2-cache and the cores communicate through shared memory. The floorplan of this chip, with an accuracy of 340 cells, is shown in Figure C. From the thermal parameters and thickness of Si and Cu layers based on data in [22], we derive the compact model parameters - conductance and capacitance matrices.

For benchmarking applications, we refer to the elaborate results collected for the UltraSPARC T1 chip as reported in [7]. For real world applications such as Web, Database, MPlayer, gcc and gzip, the utilizations of different parts of the chip are noted, and in combination with dynamic and leakage current estimates, power consumption values are computed. These computed power numbers are at the component level such as for a core or for a cache. We assume that the power consumption within such a component is homogeneous and equally distribute it among the compact model cells of that component. We compare all our results with the HotSpot simulator [11], configured with temperature-dependent compact model parameters and small time-steps.

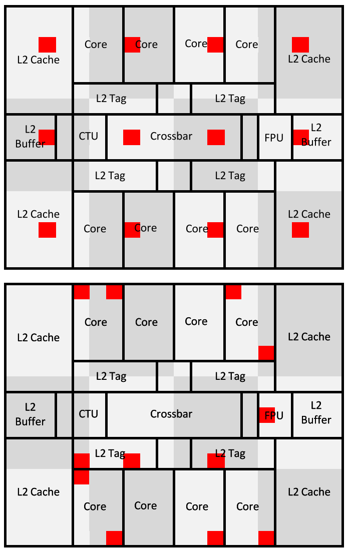

Fig. 3. Sensor layouts for `Reg` (top) and `HS` (bot) layouts

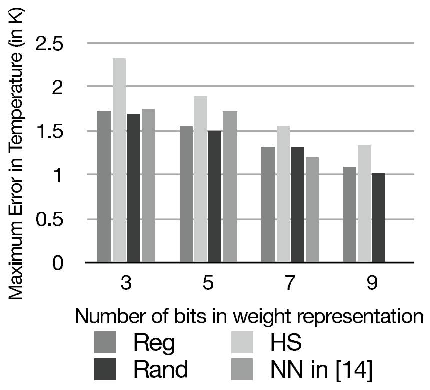

Fig. 4. Accuracy of the thermal simulation for different sensor layout configurations

B. Accuracy for different sensor layout configurations

In the first set of experiments, we study the accuracy of the extended NN simulator for different sensor layout configurations. We study the three configurations discussed earlier: `HS`, `Reg`, and `Rand`. For the Niagara chip, Fig. B shows the chosen `HS` and `Reg` layouts. In both cases, the chip is divided into 12 regions, and one sensor is placed on each region as per the configuration. For each of the choices, we vary the design parameter b - the number of bits used to represent the digitally stored weights. A higher value of b can potentially give more accurate results, but at the expense of a larger transistor count.

The results of the maximum error in temperature for different values of b are shown in Fig. B. We plot also the error in simulation for using a grid of 42 cells as reported in [14]. The results indicate that the accuracy achieved for different sensor

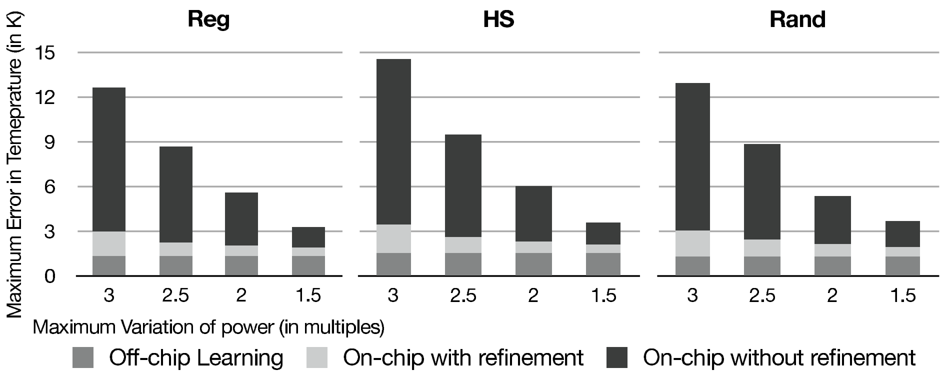

Fig. 5. Adaptability of NN to variation in leakage current

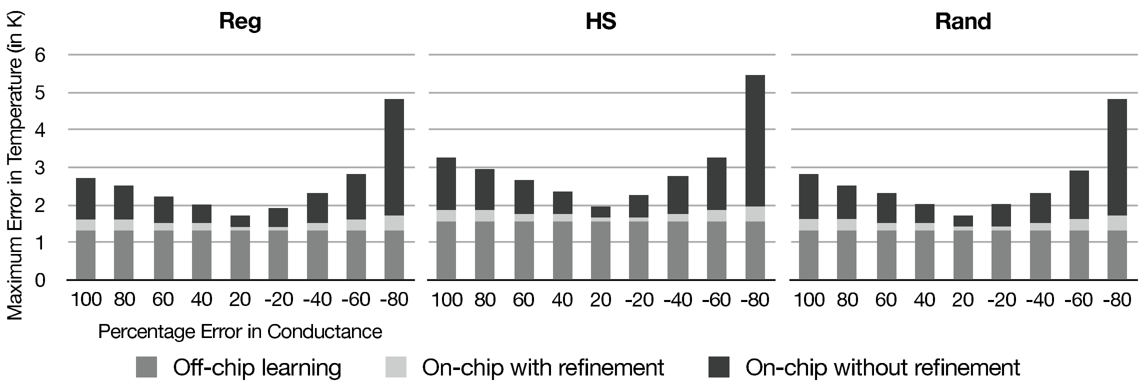

Fig. 6. Adaptability of NN to variation in conductance to ambient (G_{env})

layout configurations is comparable and in some cases better than the results presented in [14] for a uniform grid of finely separated sensors. This indicates that performing the MOR using NNs and quantization at every update according to (6) does not lead to large errors. On the contrary, the smaller size of the NN allows the learning to be more accurate than the results for a large grid as presented in [14]. Amongst the different sensor layout configurations, `HS` has the highest error while `Reg` and `Rand` perform very similarly. This is expected as the higher values of temperatures at the hotspots would lead to larger non-linearity. Thus, for an overhead of 1 byte of storage per neuron, the maximum error for all sensor layout configurations is about 1.5K.

C. Run-time refinement

In these set of experiments, we study the on-chip adaptability of NN to variations. Firstly, we consider the variation of the leakage power. Leakage-current-based power consumption is an increasingly large component of total power consumption of a chip. In [1], it has been shown that, for the 180nm CMOS technology up to 20x variation of normalized leakage power can be observed for just 30% variation in frequency. We computed the mean of the variation reported in [1] to be 7x. We conservatively study a set of cases where we progressively increase the *maximum* variation in the power consumption of the chip, between 1.5x to 3.5x. For each case, we inject random power variations, with a Gaussian distribution, on different points up to the chosen maximum variation. We perform these experiments on each of the three sensor layout configurations, with a value of $b = 9$. In all experiments, the on-chip training is performed for 100,000 iterations of HotSpot simulator running with a time step of 10μs, totaling to adaptation for 10s of real-time.

The results are shown in Fig. B, where the errors in maximum temperature are shown in stacked columns. The lowest stack of a column shows the maximum error during off-chip training, without any power variation. The next stack shows the additional error that is incurred, on introducing the power variation and *after* run-time refinement of the NN. The topmost stack indicates the additional error due to the power variation *before* the refinement and thus represents the improvement in accuracy due to on-chip adaptation. The charts indicate that significant errors in temperature, of up to 15K, are possible for expected variations in power. Further, on-chip refinement of the NN, can substantially reduce the error: by up to 90%. This trend is consistent across different sensor layout configurations.

We perform a similar set of experiments for the case of variation in the conductance to the ambient, G_{env}. We considered a general study with a varying percentage of error in G_{env}. For each case, in Fig. B, we again plot stacked columns, as described above, for each sensor layout configuration. The results indicate that the error introduced due to variation of G_{env} is of a smaller magnitude of up to 6K. On-chip adaptation again reduces the error, in all cases, to below 2K.

The main conclusion of these set of experiments is that expected variations in leakage current and thermal model can lead to unacceptably large errors in computed temperatures. Furthermore, the proposed adaptable NN can substantially reduce these errors.

V. Conclusions and future work

Power densities of chips continue to rise, leading to an ever difficult problem of effective heat dissipation. To complement hardware cooling methods, architectural-level Dynamic Thermal Management (DTM) techniques are increasingly becoming an essential part of today's systems. An important component of such techniques is the thermal trigger that initiates different response mechanisms. Intuitively, such thermal triggers perform better if they are predictive in nature. We presented a trade-off space of such predictive thermal triggers and qualitatively placed on it different triggers proposed in the literature. We identified that run-time adaptability is a key parameter which skews the otherwise simple trade-off between accuracy and efficiency. The Neural Network (NN) simulator presented in [14] provides key benefits over other thermal triggers proposed in literature.

We extended the NN simulator with two features, compatibility with an arbitrary sensor layout configuration and runtime adaptability. The extended NN simulator can simulate temperature up to 0.5s into the future with accuracies of about 1.5 K, for different sensor layout configurations. This compares favorably against the results presented for 42-cell floorplans with a much larger NN in [14]. We showed that with expected leakage current variations, the errors in computed temperature, without on-chip adaptation, can be as large as 15 K. We showed that with the proposed run-time adaptable NN simulator, these errors can be brought down by up to 90%. For variation in thermal model parameters, simulation errors can be up to 6 K, which can be brought down to about 2 K with the proposed run-time adaptable NN simulator.

In this paper we show that with 12 sensors and run-time adaptation the simulation results were accurate to up to a 2-3 K. Having more on-chip sensors would make the NN more expressive and potentially lead to improved accuracy. However, at the same time a larger NN would necessitate greater amount of training to learn the larger number of weight terms. As a future work, it would be instructive to establish this trade-off between learning time, run-time simulation accuracy and number of on-chip sensors.

Acknowledgements

This research has been partially funded by the Nano-Tera RTD project CMOSAIC (ref.123618), financed by the Swiss Confederation and scientifically evaluated by SNSF.

References

[1] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *DAC*, 2003.

[2] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, 2001.

[3] A. E. Bryson, Y.-C. Ho, and G. M. Siouris. Applied optimal control: Optimization, estimation, and control. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(6), june 1979.

[4] J.-J. Chen, S. Wang, and L. Thiele. Proactive speed scheduling for real-time tasks under thermal constraints. In *IEEE RTAS*, 2009.

[5] A. K. Coskun, D. Atienza, T. S. Rosing, T. Brunschwiler, and B. Michel. Energy-efficient variable-flow liquid cooling in 3d stacked architectures. In *DATE*, 2010.

[6] A. K. Coskun, T. S. Rosing, and K. C. Gross. Proactive temperature balancing for low cost thermal management in mpsocs. In *ICCAD*, 2008.

[7] A. K. Coskun, T. T. Rosing, K. Whisnant, and K. C. Gross. Static and dynamic temperature-aware scheduling for multiprocessor socs. *IEEE Trans. VLSI Syst.*, 16(9), 2008.

[8] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *ISCA*, 2006.

[9] G. d. M. Francessco Zanini, David Atienza. Multicore thermal management using model predictive control. In *ISCAS*, 2010.

[10] D. Grunwald, A. Klauser, S. Manne, and A. R. Pleszkun. Confidence estimation for speculation control. In *ISCA*, 1998.

[11] 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. VLSI Syst.*, 14(5), 2006.

[12] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: A 32-way multithreaded sparc processor. *IEEE Micro*, 25(2), 2005.

[13] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha. Hybdtm: a coordinated hardware-software approach for dynamic thermal management. In *DAC*, 2006.

[14] P. Kumar and D. Atienza. Neural network based on-chip thermal simulator. In *ISCAS*, 2010.

[15] K.-J. Lee, K. Skadron, and W. Huang. Analytical model for sensor placement on microprocessors. In *ICCD*, 2005.

[16] Y. Li, D. Brooks, Z. Hu, and K. Skadron. Performance, energy, and thermal considerations for smt and cmp architectures. In *HPCA*, 2005.

[17] Y. Liu, L. T. Pileggi, and A. J. Strojwas. Model order-reduction of rc(l) interconnect including variational analysis. In *DAC*, 1999.

[18] S. O. Memik, R. Mukherjee, M. Ni, and J. Long. Optimizing thermal sensor allocation for microprocessors. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(3), 2008.

[19] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. In *EuroSys*, 2006.

[20] R. Mukherjee and S. O. Memik. Systematic temperature sensor allocation and placement for microprocessors. In *DAC*, 2006.

[21] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. P. Boyd, and G. D. Micheli. Temperature-aware processor frequency assignment for mpsocs using convex optimization. In *CODES+ISSS*, 2007.

[22] M.-N. Sabry. High-precision compact-thermal models. *Components and Packaging Technologies, IEEE Transactions on*, 28(4), dec. 2005.

[23] K. Skadron, T. F. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. In *HPCA*, 2002.

[24] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *TACO*, 1(1), 2004.

[25] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *ICS*, 2003.

[26] T.-Y. Wang, Y.-M. Lee, and C. C.-P. Chen. 3d thermal-adi: an efficient chip-level transient thermal simulator. In *ISPD*, 2003.

[27] X. Wu and M. Pedram. Low power sequential circuit design by using priority encoding and clock gating. In *ISLPED*, 2000.

[28] Y. Yang, Z. P. Gu, C. Zhu, R. P. Dick, and L. Shang. Isac: Integrated space-and-time-adaptive chip-package thermal analysis. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(1), 2007.