Neural Network based On-Chip Thermal Simulator

Pratyush Kumar, David Atienza
Embedded Systems Laboratory (ESL), Ecole Polytechnique Fédérale de Lausanne (EPFL)
EPFL-STI-IEL-ESL, Station 11, 1015 Lausanne, Switzerland
{pratyush.kumar,david.atienza}@epfl.ch

***Abstract*— With increasing power densities, runtime thermal management is becoming a necessity in today's systems, especially so for highly integrated Multi-Processor Systems-on-Chip (MPSoCs). In this paper, we propose a neural network (NN) based approach to implement an on-chip thermal simulator to aid such runtime management for MPSoCs. The proposed method combines the advantage of approximating the thermal properties of the chip as a linear system with the ease of fully parallel analog implementation of NNs. We perform a case study with the Niagara UltraSPARC T1 MPSoC for real-life applications, benchmarking our results with an accurate higher order Runge-Kutta (RK4) solver, that is employed in tools such as HotSpot. Within a few gate delays, the proposed NN design can simulate temperatures of the MPSoC 500 ms into the future - corresponding to thousands of iterations of the RK4 solver, with a maximum error of 1-2 K.**

I. Introduction

The ever increasing power densities in VLSI systems are translating to higher operating temperatures and consequently, lower system reliability and poorer performance guarantees. This is especially true for highly integrated systems such as Multi-Processor Systems-on-Chip (MPSoCs). While packaging and mechanical cooling solutions are being constantly upgraded, such solutions by themselves are proving to be insufficient in facing upto the problems posed by the thermal wall. The use of runtime thermal management policies, such as clock-gating, DVS and task/thread migration, is now being widely accepted as a necessary part of today's MPSoCs [1].

For such thermal management policies, proactive control can be much more effective in avoiding thermal hazards in MPSoCs [2]. Such proactive control methods need to be able to simulate the temperatures of the chip into the future, during runtime. Another class of applications which require runtime thermal simulation is fine grain control: where thermal data is required at a finer granularity than can be provided by thermal sensors. Examples include register assignment optimization [3] and liquid coolant flow control in 3d chips [4].

In this paper, we propose a Neural Network (NN) based thermal simulator, which by virtue of its design can be easily implemented in VLSI. We perform a case study with an actual MPSoC system: the Niagara UltraSPARC T1 chip. We benchmark our NN simulator against accurate higher-order differential equation solvers, which are currently used in tools such as Hotspot [5]. The results show that within a few gate delays, with error margins of 1-2 K, the NN system can simulate 500 ms in real time. This is equivalent to thousands of iterations of the differential equation solver, which take up several seconds on a quad-core desktop computer.

This research has been partially funded by the Nano-Tera.ch NTF Project CMOSAIC (ref. 123618), which is financed by the Swiss Confederation and scientifically evaluated by SNSF.

The rest of the paper is organized as follows. In Section II, we highlight other techniques of thermal simulation that have been proposed. In Section III, we lay the theoretical foundations for using NNs for thermal simulation. We discuss a VLSI implementation in Section IV. And in Section V, we detail results of a case study with the said Niagara chip.

II. Related Work

Thermal simulation with reference to runtime management has received sufficient research attention. One line of research is to employ simple techniques such as window-based predictions to aid in proactive thermal management [2]. However, such methods are not robust enough to provide performance guarantees in all scenarios. A contrasting approach is to integrate thermal models with sophisticated optimization techniques such as convex optimization [6] and Model Predictive Control (MPC) [7]. While the former requires at design-time, detailed and accurate thermal and workload models, the later is compute intensive requiring a dedicated processing unit.

A different line of research has been to improve algorithms for thermal simulation, in terms of efficiency and accuracy, by applying concepts such as multigrid, model-reduction, conjugate heat transfer, Alternating Direction Implicit (ADI), and Green's functions. These methods again are compute intensive with complicated data-structures, and are thus more suited for design-time thermal-aware optimizations.

Thus, methods which are accurate and can provide guarantees either have software-only implementations or expensive hardware implementations. In this work, we address this trade-off with the proposed NN design, which while being based upon the accurate equations of heat flow, can be inexpensively implemented as an efficient fully parallel analog design.

III. Neural Networks for Thermal Simulation

A. Compact Model Based Thermal Simulation

Compact modelling is an important step in architecture-level thermal simulation [5], wherein the spatial and temporal evolution of temperature is captured in an equivalent electrical network. In the compact model, the Si and the Cu layers of the chip are divided into a grid of blocks (Fig. 1(a)), with each

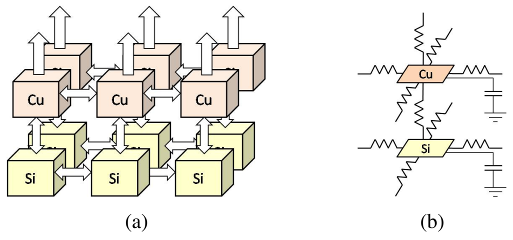

Fig. 1. Compact modelling of a chip

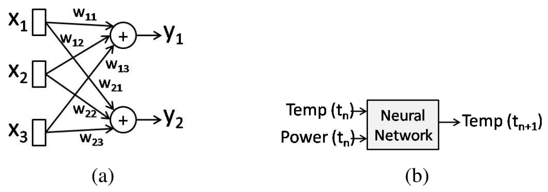

Fig. 2. Modelling thermal simulation as a Neural Network

block represented by a node with resistances to its adjacent layers and capacitances to ground (Fig. 1(b)). The Cu layer additionally is connected through interface resistances to the ambient. This network is excited by a set of current sources which correspond to power consumption in the actual chip. As per the duality between thermal and electrical systems, the node potentials obtained in the electrical network correspond to temperatures on the chip. Thermal simulation is performed by transient analysis of such a network by solving a set of first order differential equations:

$$Gx(t) + C\dot{x}(t) = i(t) \tag{1}$$

where G and C are matrices that represent the resistance and capacitance networks, respectively, $x(t)$ is the vector of node potentials and $i(t)$ is the set of injected source currents. These are most commonly solved by discretizing the time domain and iteratively solving for $x(t)$, represented in general as

$$x(t_{n+1}) = x(t_n) + hD_n \tag{2}$$

where $h = t_{n+1} - t_n$ and D_n is an iteration-dependent differential operator specific to the method employed. For all such methods, the inaccuracy rises on increasing h and thus for accurate simulations the time step h is chosen to be small enough: in the order of a hundreds of μs [8].

B. Linear Time Invariant Approximation

The thermal resistance of silicon is a function of temperature, and thus G in Eqn. 1 is a function of $x(t)$. If we, however, relax this dependence - an approximation which we evaluate with experimental results (cf. Section V.C) - then a dynamical system with state equations given by Eqn. 1 represents a Linear Time Invariant (LTI) system, for the duration of time when $i(t)$ remains constant. Changes in $i(t)$ are required at times of updated power consumption numbers. Let the power consumption values be updated such that they remain constant from $t = t_n$ to $t = t_n + \Delta t = t_{n+1}$. Then, the theory of LTI systems states that, there exist matrices A and B such that

$$x(t_{n+1}) = Ax(t_n) + Bi(t_n) \tag{3}$$

Thus, for a given circuit, neglecting the temperature dependence of resistance parameters, if we identify the matrices A and B, we can simulate the equivalent electrical network at time-steps of Δt. For MPSoCs, power values are updated once every hundreds of milliseconds. Hence Δt can be 2-3 orders of magnitude larger than h that differential equations are bounded by. Matrices A and B can be systematically identified using step-input responses. But this method would be cumbersome for large networks, which compact model based networks typically are. We propose to use NNs to learn the matrices and subsequently perform the operations of Eqn. 3.

C. Neural Network Solution

Neural networks are Multi-Input Multi-Output (MIMO) function approximators, which can be used to learn an unknown functional dependence between inputs and outputs using test data. Given our specific application of thermal simulation of MPSoCs, we restrict ourselves to a discussion of linear NNs: with a single layer and without any activation function. Consider an example 3-input 2-output linear neural network shown in Fig. 2(a). Two operators are shown: the arrows are multipliers with multiplicands shown as weights and the circles are summers. Thus, output y_i is given by

$$y_i = \sum_j w_{ij} x_j, \tag{4}$$

where w terms represent the weights. By learning appropriate weight terms, any linear dependence between the outputs and inputs can be expressed by such a NN.

Given the linear nature of Eqn. 3, we can similarly have a NN with inputs $\{x(t_n), i(t_n)\}$ and outputs $x(t_{n+1})$ (Fig. 2(b)). The weights would then appropriately model the matrices A and B. Test data for training the NN can be easily obtained from actual chip measurements or accurate computations performed by a simulator like HotSpot. While the learnt NN simulates Δt into the future in one pass through the network, it would still retain the accuracy of the method that was used to train it. Thus, A and B can be learnt without deriving them directly from the compact model parameters.

IV. VLSI Implementation

The advantages provided by the NN solution to thermal simulation are heightened by the possibility of implementing them natively in VLSI systems. In this section, we discuss an implementation of the NN system, specific to our application.

Consider a NN, where the temperature of every node of the compact model is computed by a separate neuron subcircuit so as to perform a fully parallel computation. The complexity of the interconnection required between these neurons is $O(N^2)$, where N is the number of blocks into which the chip has been divided. This can be prohibitively large, given the large size of compact models. We thus, propose a relaxation of the fully connected NN to one where interconnection is retained between output and input nodes only if the corresponding grid blocks are within a given physical distance, say r, in the chip. This approximation, though natural given the diffusive nature of heat flow, is likely to add to the inaccuracy of the

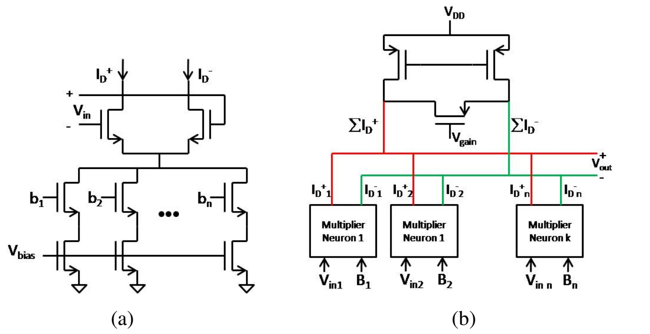

Fig. 3. Implementation of the NN subcircuit: (a) Multiplier neuron, and (b) Summer with current to voltage convertor

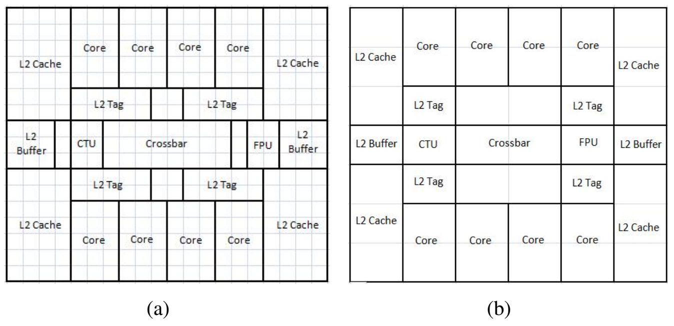

Fig. 4. Floorplan of the Niagara T1 chip divided into (a) 340 and (b) 42 cells as used in the compact modelling

obtained results. We evaluate this inaccuracy quantitatively in the section on the case study (cf. Section V).

A. Neural Network Subcircuit

As mentioned earlier, the NN subcircuit for our application is devoid of an activation function. Only linear analog multiplier neurons followed by a summer are necessary. Several analog multiplier neurons have been proposed in the literature and the topic still is being actively researched [9], [10]. The final choice of the implementation is governed by the nature of the application at hand. As would be clear from the discussion under the case study, accuracy of representing weights is a crucial factor for thermal simulation. Hence, we choose an implementation where the weights can be digitally stored. An added advantage is that, if the digital weights are made software programmable, then software-aided learning can be performed on-chip. To provide the desired high simulation speed, the multiplication should be analog in nature.

Given these considerations we choose to use the multiplier neuron, shown in Fig. 3(a), as presented in [11]. The weight is digitally expressed by the string of bits $b_n \dots b_0$. The W/L ratio of the transistors are suitably designed to provide a binary-weighted current source array. Such neuron multipliers are connected together to form a summer with a differential current-to-voltage convertor as shown in Fig. 3(b). Thus, as in Eqn. 4, the output voltage, for some constant K, is given by

$$V_{out} = K \sum_j \Big(\sum_i b_{ij} 2^i \Big) V_{in_j} \tag{5}$$

V. Case Study

A. Target System and Benchmark Applications

Our case study is based on the 8-core UltraSPARC T1 (Niagara-1) architecture from Sun Microsystems [12]. The floorplan of this chip in two accuracies for the compact model: 340 and 42 cells, is shown in Fig. 4. We derive the thermal parameters and thickness of Si and Cu layers based on [13]. For benchmarking applications we have chosen several real-life applications which are run on the UltraSPARC T1 chip and the utilizations of the cores are noted as reported in [14]. From these utilization numbers we derived the power traces based on the average power values reported in [12].

B. Training the Neural Network

We use the fourth-order Runge-Kutta (RK4) differential equation solver, with a small time-step of $h = 100\mu s$ to generate accurate temperature profiles to train the NN, with the backpropagation learning algorithm [15]. We train the NN every 5000 iterations of the RK4 solver. Thus, the NN is trained to predict the temperatures 500 ms into the future. In [2], it has been shown that for the UltraSPRAC T1 system, proactive control with predictions of 500 ms can reduce hotspots by 60% over reactionary policies.

An important issue with the analog implementation of NNs is the quantization error in translating real valued weights to corresponding parameters of the analog circuit. The chip-in-the-loop technique [16] is a method to alleviate this problem. With this approach, learning of the NN is performed ensuring that at all times the parameters of the NN are such that an accurate translation can be made to an analog circuit. From the circuit shown in Fig. 3(a), it is clear that the quantization is governed by the number of bits, say b, used to represent the digital weights: the least normalized unit is 2^{-b}. In the training phase, at the end of every training epoch, we ensure that weights are rounded to this normalized unit.

C. Results

After training the NN, we benchmark it against the iterative RK4 solver that was used to train the NN. We note that the RK4 solver is used in tools such as HotSpot and factors the temperature dependence of the resistance matrix. Comparing our NN method with it would indicate the inaccuracies introduced by the LTI assumption and due to the finite values of b and r. We quantify this inaccuracy by $E_{\max}$ defined as the absolute value of the temperature difference between the RK4 solver and the trained NN, maximized across all blocks of the compact model over a large number of simulation iterations.

We study three parameters of the design: a) accuracy of the floorplan as used in the compact model, b) r - the maximum distance between cells with interacting neurons, and c) b - the number of bits used to represent the digital weights. The accuracy of the floorplan would be dependent on the application. Most proactive control techniques are still at the core level and representing the floorplan at a coarse level would suffice functionally. Fine grain control however, would

TABLE I

E AND TRANSISTOR COUNT FOR VARIOUS DESIGN POINTS

r	b	42 Cells		340 Cells	
		Emax (in K)	Trans. Cnt. (x1000)	Emax (in K)	Trans. Cnt. (x1000)
Full	3	1.7473	28	2.3896	1850
	5	1.7214	42	2.2428	2775
	7	1.2311	56	1.7912	3699
	∞	0.9969	-	1.1213	-
5	3	4.4683	8	8.8695	69
	5	2.8861	12	6.1024	102
	7	1.3322	17	2.0436	137
	∞	1.2433	-	1.6790	-
4	3	7.3592	5	14.6799	44
	5	3.1695	8	8.0665	66
	7	1.3920	10	2.5806	88
	∞	1.3822	-	1.9787	-
3	3	12.6393	3	19.7790	25
	5	4.3556	5	11.9544	37
	7	2.0842	6	4.3181	49
	∞	1.7888	-	3.1191	-

require a higher accuracy of the floorplan. As shown earlier in Fig. 4, we choose to divide the floorplan into 42 and 340 cells, to be representative of the above two scenarios.

The finer the floorplan, the more the number of temperatures to be predicted, and lower is the expected accuracy for a given fixed amount of training. Further, when either of the two design parameters, r and b, is increased, a higher accuracy is expected, but at a larger on-chip area cost. To study the dependence of the accuracy on these paraemters, we vary the parameters and for each combination, train and benchmark a separate NN. The results as given by $E_{\max}$ and transistor counts necessary to implement the neuron multipliers (Fig. 3(a)) are shown in Table I. The results are also shown in the scatter plot in Fig. 5. We mark one pareto-optimal design point for either accuracy of floorplan (labelled A and B), to aid our discussion. These design points restrict the maximum error over all cells and simulation iterations to within 1-2 K: a 3-5% margin of the typical operating temperature ranges of MPSoCs. These designs have transistor counts of about 10 and 100 thousands, for the 42 and 340 cell floorplans, resepectively, which are acceptably small fractions of the transistor count for the full Niagara chip - reported at 279 million. Thus, with small error margins and acceptable area cost, within a few gate delays, the NN designs can thermally simulate the MPSoC for 500 ms. For an iterative thermal simulator like HotSpot, this is equivalent to thousands of iterations of the RK4 solver, which on average required several seconds, on a quad-core desktop computer.

VI. CONCLUSIONS AND PROSPECTS

The proposed neural network based solution to thermally simulate on-chip combines the advantages of applying the LTI approximation to accelerate thermal simulation with those of NNs: learning with test-data and effectiveness of native VLSI implementation. For the Niagara UltraSPARC T1 chip, we studied a specific neuron circuit, identified design parameters and experimented with several NN designs to quantify the tradeoff of implementation cost and accuracy. The results

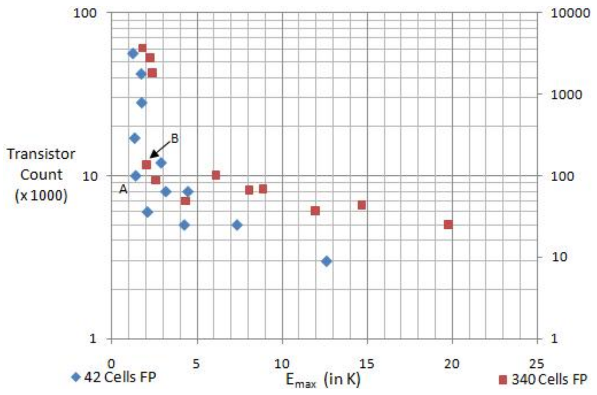

Fig. 5. Plotting of various design points for pareto-optimizations

indicate that with a carefully chosen set of design parameters, an accuracy of within about 1-2 K, for thermal simulating the MPSoC for 500 ms in real time, can be achieved with a design of about 10-100 thousand transistors.

This work strongly motivates the integration of NN based simulators into an all-hardware sensor-simulator-controller-actuator loop. Such a system can potentially react to impending thermal hazards much quicker than existing methods which rely on software constructs. Further, the reconfigurable nature of NNs opens the unique prospect of on-chip reconfiguration, which can negate modelling errors, account for process variations and adapt to changing workloads.

REFERENCES

[1] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration", ISCA, 2006, pp. 78-88.
[2] A. K. Coskun, et al: "Proactive temperature balancing for low cost thermal management in MPSoCs", *ICCAD*, 2008
[3] D. Atienza, et al, "Joint Hardware-Software Leakage Minimization Approach for the Register File of VLIW Embedded Architectures", *Integration - The VLSI journal*, 41(1):38-48, 2008
[4] A. K. Coskun, et al, "Modeling and Dynamic Management of 3D Multicore Systems with Liquid Cooling", *VLSI-SoC*, vol. 1, 2009
[5] W. Huang et. al, " HotSpot: a compact thermal modeling methodology for early-stage VLSI design", *Trans. on VLSI*, vol. 14, 2006
[6] S. Murali, et al, " Temperature-aware processor frequency assignment for MPSoCs using convex optimization", *DAC*, 2008.
[7] F. Zanini, et al, "Multicore Thermal Management with Model Predictive Control", *ECTTD*, vol. 1, 2009.
[8] F. Zanini, et al, "Optimal Multi-Processor SoC Thermal Simulation via Adaptive Differential Equation Solvers", VLSI-SoC, vol. 1, 2009
[9] C. Mead, M. Ismail, "Analog VLSI implementation of neural systems", Springer, 1989
[10] M. Valle, "Analog VLSI Implementation of Artificial Neural Networks with Supervised On-Chip Learning", *Analog Integrated Circuits and Signal Processing*, 3, pp. 263-287, 2002
[11] P. Hollis, J. Paulos, "Artificial neural networks using MOS analog multipliers", *Journal of Solid-State Circuits*, 1990
[12] P. Kongetira et al., "Niagara: A 32-way multithreaded SPARC processor", *IEEE Micro*, 2005
[13] M. N. Sabry, "High-precision compact-thermal models", *Trans. on Components and Packaging Tech.*, 2005
[14] A. K. Coskun, et al., "Static and Dynamic Temperature-Aware Scheduling for Multiprocessor SoCs", *IEEE Trans. on VLSI*, vol.16 no.9, pp. 1127-1140, Sept. 2008
[15] P. Baldi, K. Hornik, "Learning in Linear Neural Networks: a Survey", *Trans. on Neural Networks*, vol. 6, 1995
[16] A. Annema, "Feed-forward Neural Networks", Kluwer Academic Publishers, 1995