Measuring and Predicting Temperature Distributions on FPGAs at Run-Time

Markus Happe, Andreas Agne and Christian Plessl
Computer Engineering Department
University of Paderborn, Germany
Email: {markus.happe, agne, christian.plessl}@uni-paderborn.de

***Abstract*—In the next decades, hybrid multi-cores will be the predominant architecture for reconfigurable FPGA-based systems. Temperature-aware thread mapping strategies are key for providing dependability in such systems. These strategies rely on measuring the temperature distribution and predicting the thermal behavior of the system when there are changes to the hardware and software running on the FPGA. While there are a number of tools that use thermal models to predict temperature distributions at design time, these tools lack the flexibility to autonomously adjust to changing FPGA configurations.**

To address this problem we propose a temperature-aware system that empowers FPGA-based reconfigurable multi-cores to autonomously predict the on-chip temperature distribution for pro-active thread remapping. Our system obtains temperature measurements through a self-calibrating grid of sensors and uses area constrained heat-generating circuits in order to generate spatial and temporal temperature gradients. The generated temperature variations are then used to learn the free parameters of the system's thermal model. The system thus acquires an understanding of its own thermal characteristics.

We implemented an FPGA system containing a net of 144 temperature sensors on a Xilinx Virtex-6 LX240T FPGA that is aware of its thermal model. Finally, we show that the temperature predictions vary less than 0.72 degree C on average compared to the measured temperature distributions at run-time.

***Keywords*-temperature measurement, thread mapping, thermal model, temperature-awareness, ring-oscillator, FPGA**

I. Introduction

Thermal management gains importance for FPGA-based embedded systems, since thermal effects dramatically increase for shrinking device structures. High temperatures influence the switching speed of transistors, which can lead to soft (timing) errors, and contribute to the premature occurrence of hard errors [1]. Over the last years, several dynamic thermal management techniques were discussed, which aim at avoiding hot spots (spots on the chip exceeding certain temperature thresholds) and at balancing the on-chip temperature.

One popular approach to enable thermal management on embedded multi-core systems is thermal thread mapping. For instance, [2], [3] developed thread mapping strategies for FPGA-based multi-processor systems-on-chip. Using thermal thread mapping strategies, the operating system maps threads to cores such that currently hot cores run with reduced thermal stress, while currently cold cores have to bear a higher level of thermal stress. We believe that these mapping strategies are not limited to multi-processor systems, but can be extended to heterogeneous multi-core systems, e.g. ReconOS [4], containing processors that execute software threads and reconfigurable hardware slots that execute hardware threads. However, predicting accurate temperature distributions of different mapping scenarios (based on the current temperature distribution and thread mapping) on-line is still an open research challenge. This challenge needs to be solved, so that thermal thread mapping strategies are able to select the optimal mapping variant and, hence, the the number of thread migrations can be significantly reduced compared to trial and error strategies.

To meet this challenge, the system, on the one hand, needs to measure the current temperature distribution at run-time. While this is already possible using a sensor net based on ring oscillators [5], these sensors have to be calibrated manually using external devices, e.g. a temperature-controlled oven, beforehand. On the other hand, the system, requires a thermal model to predict or simulate the effect of thread remapping. Today's thermal models of VLSI designs, unfortunately, require knowledge of the chip structure and the material properties which have to be provided by the system designer. Furthermore, the complexity of today's thermal models, e.g. HotSpot [6], prevents their application on embedded devices. To overcome these shortcomings, we propose a system that is temperature-aware in the sense, that it is able to calibrate its sensors and to construct its internal thermal model without outside intervention. Thus, such a system enables autonomous thermal thread mapping.

Following our approach, the system first calibrates its sensors and then learns its internal thermal model autonomously in an initial testing phase, before the system starts execution. As precondition, we assume that the FPGA contains an built-in pre-calibrated thermal diode. We fill the FPGA with local heat-generating circuits (heaters) and place a net of ring oscillator-based temperature sensors on top of it. In a first step, the system uses all local heaters to globally heat up the chip. The sensors are calibrated to the temperature readings of the thermal diode while heating. In a second step, the system generates spatial gradients by activating only some of the local heaters. Using a learning algorithm, the system determines the parameters of its internal thermal model. This enables modern FPGA-based systems to measure and predict

Published in: *Proc. Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig)*
 DOI: 10.1109/ReConFig.2011.59

temperature distributions autonomously which forms a basis for later thermal thread mapping strategies on (heterogeneous) multi-cores.

As novel contribution, we propose a temperature-aware system that is able to measure and predict temperature distributions on-chip without any knowledge about the chip structure or the material properties of the FPGA. Finally, our proposed sensor calibration technique replaces the extensive manual sensor calibration using external devices, e.g. a temperature-controlled oven [5], [7], [8] or an infrared camera [9], by self-calibration using local heaters and a built-in thermal diode.

The remainder of the paper is structured as follows. We discuss related work on temperature sensors, temperature models and temperature simulations for FPGAs in Section II. We introduce our system architecture that can measure temperature distributions and generate temporal and spatial thermal gradients on-chip in Section III. Section IV presents our proposed thermal model and a learning algorithm that identifies the system's thermal model parameters at run-time. The learned parameters can be used to predict future temperature distributions, e.g. after thread mapping. In Section V, experimental results for on-chip sensor calibration, thermal model parameter learning and temperature predictions are presented and analyzed for an Xilinx Virtex-6 LX240T FPGA. Finally, Section VI concludes the paper.

II. Related Work

This section presents related work in temperature measurements, temperature models and temperature simulations in FPGA-based systems.

A. Measuring Temperatures on FPGAs

In the last decade, several works [5], [7] used ring oscillators combined with counters to design temperature sensors on FPGAs. More recently, [8] designed a net of such sensors on a Xilinx Virtex-5 and showed that ring oscillator-based sensors can furthermore be used to measure leakage, delay and dynamic power. Similar to [5], [7], [8], we use a ring oscillator combined with a counter as temperature sensor. Unlike [5], [7]–[9], we do not calibrate the sensors using a temperature-controlled oven or an infrared camera. Instead, our system self-calibrates its sensors using local heat-generating cores and an internal thermal diode which can be found in many modern FPGA devices, e.g. Xilinx Virtex-5 and Virtex-6 FPGAs.

B. Temperature models and simulations

HotSpot [6], [10] is a widely-used tool for temperature simulations of VLSI designs, e.g. FPGAs. HotSpot provides a compact thermal modeling methodology that makes use of the duality between thermal and electrical phenomena. Thus, it defines a multi-layered RC-network to compute the heat flow on the chip. The model consists of multiple layers like heat sink, heat spreader, thermal interface material, silicon bulk, interconnect layer, etc. Each layer is partitioned into blocks. Each block has a thermal capacitance (heat absorption capability) and is connected to other blocks with lateral and vertical thermal resistances. Many model parameters of the RC-network can be defined by the material properties of the considered layer. In [7], the authors show that the temperature distributions on FPGA systems that are measured using ring oscillators closely match the simulated temperature distributions using HotSpot. Although, HotSpot provides a compact thermal model methodology for fast temperature simulations, it is infeasible to use it on embedded systems due to its high level of detail. Thus, we use a thermal model with only two layers, trading accuracy for performance. Unlike [6], [10], we assume that our system cannot define the model parameter using material properties provided by the manufacturer but learns them at run-time.

III. Measuring Temperature Distributions

A. Temperature Sensor

A ring oscillator contains an odd number of inverters as depicted in Figure 1 whose output oscillates between '0' and '1' at some frequency. The frequency of the ring oscillator changes almost proportional to the temperature, since the switching speed of transistors is directly influenced by the temperature which was shown in [5]. Before a measurement is done, the oscillator 'tunes in' for a short amount of time. Then, a counter counts the oscillations for a fixed amount of time to define the current frequency of the oscillator which can be translated into temperatures. We enable our ring oscillator-based temperature sensor for 2^{13} system clock cycles before measuring. Then, we measure for 2^{17} system clock cycles to reduce the impact of the system's noise. Note, that the ring oscillators have to be designed such that their frequencies is at most half of the counter's clock frequency in order to provide reliable sampling. Ring oscillators containing 11 inverters showed good temperature sensitivity when the inverters were mapped to different slices.

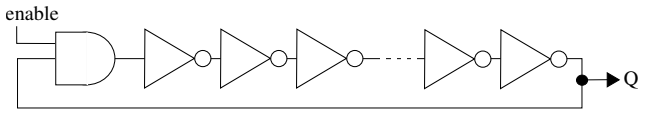


Figure 1. Ring oscillator with an odd number of inverters.

B. Heat-generating Circuits

As our temperature-aware system needs to learn its internal thermal model, it has to be able to generate spatial thermal gradients. Thus, we developed local heat-generating circuits (heaters) and map them on the FPGA using area constraints. Figure 3(b) shows an exemplary positioning of such heaters on an FPGA. All heaters are connected

to a system bus and can be activated independently from a MicroBlaze processor. When activated, a heater toggles 10,000 flip flops at each clock cycle to produce heat. Using an Virtex-6 FPGA, we were able to generate spatial thermal gradients up to 6.5°C.

C. Sensor Calibration

To translate the measured frequencies of a ring oscillator-based sensor to temperatures, the system needs to calibrate the sensor. To apply our approach, the FPGA needs to contain a pre-calibrated built-in thermal diode which the system uses to calibrate the sensors. Such thermal diodes can be found in Virtex-5 and Virtex-6 FPGAs. According to [5], [7], the correlation between measured frequencies and temperatures is almost linear. Thus, we propose that the system calibrates a sensor by mapping distinct sensor frequency measurements to the corresponding temperature measurements of the thermal diode. For a sensor calibration, the on-chip temperature distribution should be balanced. Hence, the system only measures twice, (1) where all heaters are deactivated and (2) where all heaters are activated. The system generates a linear frequency-temperature transformation function $T(f)$ that is defined by the measurements. Since sensors might be routed differently, each sensor needs its own translation function. Once the individual translation functions are defined, the system can measure temperature distributions on the chip.

D. Sensor Net

In order to get a good sensor coverage of the FPGA, we propose to partition the FPGA into an regular grid of tiles and place a sensor at the center of each tile. Thus, the sensors form a sensor net where all sensors are connected to the same monitoring unit. The monitoring unit is connected to a MicroBlaze CPU that controls not only the heaters but also the sensors using the PLB bus. When a measurement is triggered by software each sensor performs a measurement as described in Section III-A. The individual counter values can be accessed by the MicroBlaze processor using the PLB bus. These counter values can then be translated to temperatures using the calibrated translation functions $T(f)$. An exemplary sensor net can be seen in Figure 3(a).

IV. Predicting Temperature Distributions

The purpose of our thermal model lies not so much in accurately modeling physical reality given a set of parameters, as it is done for instance in design-time thermal analysis [6]. Instead we focus on an efficiently computable model with a small number of free parameters that can be easily found by an on-line learning algorithm.

A. Model Layout

In order to model heat flow and temperature distributions we make use of the well known duality between thermal models and RC-networks. Our model consists of two vertically arranged layers, L_0 and L_1. Each layer is a regular grid of nodes of width w and height h, that matches the layout of the temperature sensor grid. Each node i is identified by a coordinate $p(i) = (x, y)$, and the number $l \in \{0, 1\}$ of its layer. A capacity $C(i)$ is assigned to each node. Nodes within a layer l are connected by the resistance $R_{x,l}$, for nodes that are neighbors in the x-coordinate and $R_{y,l}$ for neighbors in the y-coordinate. Nodes of different layers are vertically connected through the resistances $R_v(i_0, i_1)$ with $i_0 \in L_0$ and $i_1 \in L_1$ with $p(i_0) = p(i_1)$. The nodes i_1 in layer 1 are connected to a heat sink of temperature T_s with the resistances $R_s(i_1)$ while the nodes i_0 in layer 0 are connected to heat sources $I(i_0)$. Figure 2 illustrates the model layout.

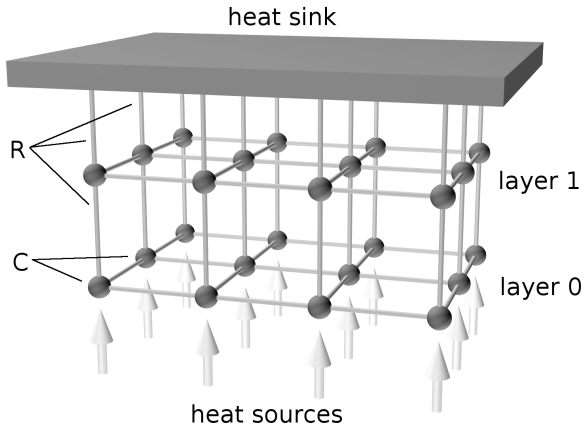


Figure 2. Thermal model: The nodes are arranged in two layers where each layer is a regular grid of nodes. Layer 0 receives heat input while layer 1 is connected to the heat sink.

In this model the heat flow $I_n(i)$ to a node i from its neighbors $N(i)$ is given as

$$I_n(i) = \sum_{j \in N(i)} \frac{T(j) - T(i)}{R(i, j)}$$

where $T(i)$ is the current temperature of node i and $R(i, j)$ is the resistance between the nodes i and j. The flow I_{sink} to the heat sink is:

$$I_{sink}(i) = \frac{T_s - T(i)}{R_s(i)}$$

For small time intervals Δt the temperature change ΔT of a node i can be approximated as

$$\Delta T_i = \frac{I_n(i) + I_{source}(i) + I_{sink}(i)}{C(i)} \Delta t \qquad (1)$$

where $I_{source}(i)$ is the heat input generated by circuits on the FPGA. Only layer L_0 is connected to sources and only

layer L_1 is connected to sinks. Therefore, $I_{source}(i_1) = 0$ and $I_{sink}(i_0) = 0$ for all $i_0 \in L_0$ and $i_1 \in L_1$.

Using a two layer model is a compromise between prediction accuracy and computational efficiency: A one layer model cannot generate the temporal gradients we observed while three or more layers do only marginally improve predictions and at the same time generate a greater workload for the learning algorithm because of the increased number of free parameters.

B. Learning Parameters

The model we propose contains a set P of free parameters that have to be learned by the system in order to make useful predictions (see Table III). In order to evaluate how good a parameter set P is, we first measure a time series of temperature distributions, then we run a simulation of our model using P and compare the resulting temperature distribution $T_s(P, t_i, j)$ at each time step t_i and at each node $j \in L_0$ with the measurement $T_m(t_i, j)$. The goal is to find a parameter set P that minimizes the mean square error *mse* of the simulation given by:

$$mse(P) = \frac{1}{|N||M|} \sum_{i \in M} \sum_{j \in N} \left(T_s(P, t_i, j) - T_m(t_i, j)\right)^2$$

where N is the set of layer-0 nodes and M is the set of time indices at which measurements were taken. We do this in two steps: First, we generate a spatially uniform temporal temperature gradient by activating all heaters at the same time. Since this leads to a spatially uniform temperature distribution, there is no heat flow I_n between neighboring nodes on the same layer. This allows us to learn the parameters that determine the vertical heat flow through our model independent of the lateral components. The learning itself is done by randomized hill climbing, where the objective function to be minimized is $mse(P)$. The algorithm starts from an initial solution $P = P_{init}$ and through random variation of P tries to improve $mse(P)$. Over a pre-defined number of iterations the random variations become smaller until the algorithm terminates in or close to a minimum.

In the second step, we generate spatial gradients by selectively activating the heaters to the top, bottom, left and to the right. This is mainly used to find values for the lateral resistances $R_{x,l}$ and $R_{y,l}$.

C. Predicting Temperature Distribution

Using the previously learned thermal model, the temperature-aware system can predict future temperature distributions at run-time: The system initializes the tile temperatures of layer 0 with the current measurements of the temperature sensors. If some of the tiles do not contain temperature sensors, their temperatures can be gauged by neighboring tiles that contain sensors. The temperatures of the tiles in layer 1 can then be initialized so that the system is in thermal equilibrium. For temperature prediction, the changes in temperature have to be updated iteratively for all tiles using Equation 1. Continuous updates of the tile temperatures for small Δt values lead to accurate temperature predictions of arbitrary length.

In the case of a reconfigurable multithreaded HW/SW system, each thread may be mapped to a CPU or to a digital circuit implemented on the FPGA. Each thread has a thermal footprint in the form of heat sources $I_{\text{source}}(i)$ that depends on where it is running and on its current workload. These thermal footprints have to be known in advance in order to predict the system's future temperature distribution.

We propose that the temperature-aware system learns the $I_{\text{source}}(i)$ parameters of different threads at run-time and during thread execution. In the beginning, this implies that the system has to explore the effects of different thread mappings. If an entire thread schedule should be predicted, the system has to change the heat sources of affected tiles for the specific points in time. An example can be seen in Figure 5 where the activation of the local heaters change at run-time.

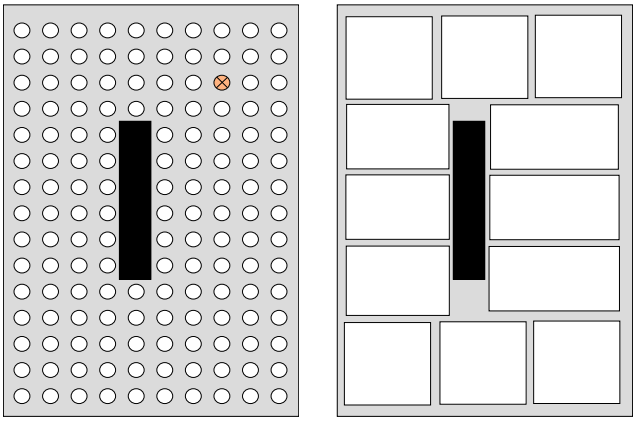
V. Experimental Results

A. FPGA Setup

For experimental measurements, we used the Xilinx Virtex-6 LX240T FPGA ML605 Evaluation Kit that contains a pre-calibrated internal thermal diode that can be accessed inside the FPGA using the dedicated system monitor hard macro. We partitioned the FPGA containing 160x239 slices into an regular grid of 10x15 tiles where each tile contains a temperature sensor at the center. For our experiments this setup proved to be a good compromise between grid resolution and FPGA resource usage. On this FPGA, there is a central region which is not reconfigurable so that we could not place temperature sensors for 6 tiles in this region. Figure 3(a) shows our sensor net. We implemented 12 heat-generating circuits and constrained them to disjunct areas that can be seen in Figure 3(b). This layout was chosen because on one hand it enables us to heat up the left, right, upper and lower sides of the FPGA independently and on the other hand the size and shape of the heater regions resemble that of possible hardware threads that may run on the FPGA.

B. Calibrate Sensor Net

In order to calibrate the sensors, the system must first be in thermal equilibrium, which we assume is reached when the thermal diode does not vary more than 0.3°C in a time interval of 20 seconds. At this point the system makes a temperature measurement for each sensor and stores the number of oscillations for a fixed time interval of 2^{17} clock cycles as well as the measured temperature of the thermal diode. In a second step, the FPGA activates all heat-generating circuits and again waits for thermal equilibrium. At this point, the system makes a second temperature measurement



(a) sensor net (b) heaters

Figure 3. FPGA setup: (a) sensor net and (b) regional heaters mapped on a Virtex-6 LX240T ML605 FPGA. The black box in the center is not reconfigurable.

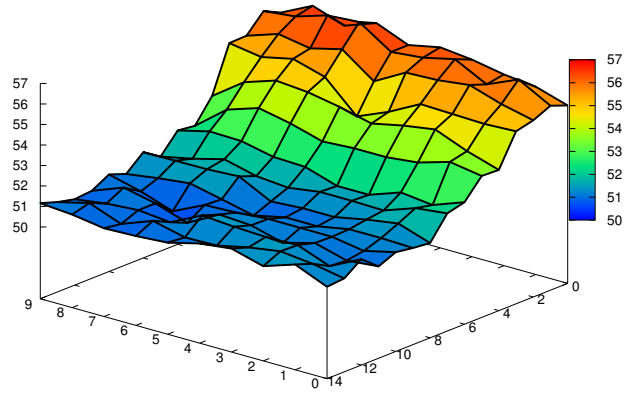


Figure 4. Measuring temperature distribution (in °C) on an Virtex-6 FPGA where the top five heat-generating cores are activated.

for each sensor and computes a linear function mapping sensor readings to temperature. The sensor self-calibration takes about 3 to 4 minutes in our experiments.

Figure 4 depicts an exemplary temperature distribution that was measured after self-calibration. In all our experiments, the fan of the FPGA's cooling element is active.

C. Learn Thermal Model Parameters

After sensor calibration, the temperature model parameters are learned. Therefore, the system creates a 12-minute test scenario, see Table I, where the system creates spatial and temporal temperature gradients. In a first step, the system activates all heaters for 2 minutes, before it deactivates them again. In a second step, it activates the local heaters on each side of the FPGA for 1 minute, before it deactivates them again for 1 minute. The temperature reading of a sensor (marked in Figure 3(a)) can be seen in Figure 5. The system performs measurements on the entire sensor net each second and stores all temperature readings in main memory.

Table I
12-MINUTE LEARNING SCENARIO

time (min.)	description
1-3	all local heaters are activated
4-5	top five local heaters are activated
6-7	bottom five local heaters are activated
8-9	five local heaters to the right are activated
10-11	five local heaters to the left are activated
other	all local heaters are deactivated

Then, randomized hill climbing–see Section IV-B–is applied for the measurement data. Table III shows the initial set P_{init} of the free parameters for the learning algorithm that were defined manually. Table II defines the improvement of the average root mean square error (rmse) between measurement and simulation data while the parameters are learned for both stages. The temperatures of the tiles in layer 0 are initialized with the first measurement. For this scenario the number of heat sources $I_{\text{source}}(i)$ is limited to the cases where the local heater that covers tile i is activated, $I_{\text{on}}(i)$, and where the local heater is deactivated, $I_{\text{off}}(i)$.

Table II
LEARNING PROGRESS OF THE RANDOMIZED HILL CLIMBING ALGORITHM

stage	rmse in (°C)
initial	3.256703
stage 1	0.773082
stage 2	0.719692

Here, the learning algorithm is executed on a MicroBlaze processor clocked at 100 MHz. Learning the thermal model parameters at run-time takes between 50 and 60 minutes depending on the input data in our experiments. The learned (averaged) temperature model parameters $\mathrm{P}_{\text{learned}}$ are listed in Table III.

Table III
INITIAL AND LEARNED TEMPERATURE MODEL PARAMETER

param.	P_{init}	$\mathrm{P}_{\text{learned}}$	param.	P_{init}	$\mathrm{P}_{\text{learned}}$
$R_v(i)$	100	101.07	$C(i), i \in L_0$	0.001	0.00099
$R_s(i)$	33.333	35.164	$C(i), i \in L_1$	1.5	1.51013
$R_{x,0}$	150	151.87	$I_{\text{on}}(i)$	0.25	0.22636
$R_{y,0}$	150	148.47	$I_{\text{off}}(i)$	0.15	0.17421
$R_{x,1}$	0.0667	0.0682	T_s	25	25.2999
$R_{y,1}$	0.0667	0.0681			

D. Compare Predictions to Measurements

Figure 5 compares the temperature measurements with the temperature predictions according to the learned temperature model for an exemplary sensor. It can be seen that the temperature predictions closely match the measured temperatures. The average prediction error for all sensors is 0.72°C for this 12-minute scenario. Predicting the entire

scenario on a MicroBlaze processor clocked at 100 MHz at run-time takes 99.5 seconds. We used a time resolution of $\Delta t = 0.02$ seconds for a prediction step. The prediction time can be reduced if the time resolution is reduced, but this has negative effects on the prediction accuracy.

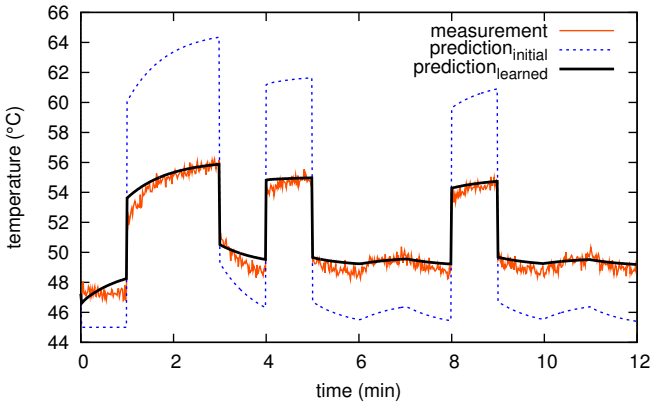


Figure 5. Temperature difference between measured and predicted temperature over the entire scenario sample set (defined in Table I) for an exemplary sensor. The sensor position is marked in Figure 3(a)

VI. Conclusion

In this paper, we have proposed a temperature-aware design for FPGA-based systems that can measure and predict temperature distributions at run-time. For hybrid multi-cores, our approach forms a basis for future thermal thread mapping strategies that can predict temperature distributions of different thread mappings at run-time. These systems can minimize the number of migrations compared to trial and error strategies, since they can select the optimal thread mapping to balance the on-chip temperature.

For measuring temperature distributions, we have presented a novel self-calibration technique for FPGAs that calibrates a ring oscillator-based sensor net internally using a pre-calibrated internal thermal diode and local heat-generating circuits instead of external devices. In our experiments, the system was able to self-calibrate its sensors in 3-4 minutes and measured spatial thermal gradients up to 6.5°C and temporal thermal gradients up to 8°C. Furthermore, we have introduced a thermal model where the system learns the model parameters following a 2-stage randomized hill climbing algorithm at run-time. Using the learned thermal model, the system is able to predict future temperature distributions with an average root mean square error of 0.72°C in a measured temperature range of 8°C.

For future work, we want to quantify the calibration error of our approach using temperature measurements obtained by an infrared camera. Then, we plan to extend the proposed temperature-aware approaches to hybrid multi-cores containing processors and reconfigurable hardware modules. The temperature sensor map will help us to identify hot spots which shall be avoided using thread mapping. Furthermore, we want to study at which quality an FPGA-based multi-core system can predict the temperature distributions of possible thread remappings at run-time. Finally, we plan to develop and analyze different thermal thread mapping techniques that benefit from these temperature predictions and apply them to a real-world case study.

Acknowledgment

This work was partly supported by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500) and by the International Graduate School of Dynamic Intelligent Systems. The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement n°257906.

References

[1] S. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE MICRO*, pp. 10–16, Nov./Dec. 2005.

[2] F. Mulas, D. Atienza, A. Acquaviva, S. Carta, L. Benini, and G. DeMicheli, "Thermal Balancing Policy for Multiprocessor Stream Computing Platforms," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 28, no. 12, pp. 1870–1882, 2009.

[3] T. Ebi, M. A. A. Faruque, and J. Henkel, "TAPE: Thermal-Aware Agent-Based Power Economy for Multi/Many-Core Architectures," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2009.

[4] E. Lübbers and M. Platzner, "ReconOS: Multithreaded Programming for Reconfigurable Computers," *ACM Transactions in Embedded Computing Systemy (TECS)*, vol. 9, 2009.

[5] S. Lopez-Buedo, J. Garrido, and E. I. Boemo, "Dynamically Inserting, Operating, and Eliminating Thermal Sensors of FPGA-Based Systems," *IEEE Transactions on Components on Packaging Technologies*, vol. 25, no. 4, pp. 561–566, 2002.

[6] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, and K. S. M. R. Stan, "HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, 2006.

[7] S. Velusamy, W. Huang, J. Lach, M. Stan, and K. Skadron, "Monitoring Temperature in FPGA based SoCs," in *Proceedings of the IEEE Int. Conf. on Computer Design*, 2005.

[8] K. M. Zick and J. P. Hayes, "On-Line Sensing for Healthier FPGA Systems," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2010.

[9] A. N. Nowroz and S. Reda, "Thermal and Power Characterization of Field-Programmable Gate Arrays," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2011.

[10] W. Huang, K. Skadron, S. Gurumurthi, R. J. Ribando, and M. R. Stan, "Differentiating the Roles of IR Measurement and Simulation for Power and Temperature-Aware Design," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, 2009.