Toolset for Measuring Thermal Behavior of FPGA Devices

Paweł Weber[1], Maciej Zagrabski[1], Bartosz Wojciechowski*[1], Krzysztof S. Berezowski[1], Maciej Nikodem[1], Krzysztof Kępa[2],

[1] Institute of Computer Engineering, Control and Robotics, Wrocław University of Technology, Wrocław, Poland
[2] Virginia Bioinformatics Institute and Configurable Computing Lab, Virginia Tech, Blacksburg, VA, USA

* Corresponding Author: bartosz.wojciechowski@pwr.wroc.pl, +48 71 3202873

Abstract

In this work, we present a toolset suitable for the analysis of thermal behavior of FPGA devices. The toolset allows for the automatic synthesis of a unified temperature measurement and heat generation infrastructure combined with the necessary control structures and communication interfaces. The tools insert temperature sensors and heaters based on ring oscillators through the low level manipulations on Xilinx Design Language descriptions of FPGA resource allocations. The purpose of the kit is to support rapid setup of thermal experiments by providing basic heating and sensing elements with verified properties as well as an area-optimised IP core for control of the experiments.

1 Introduction

In recent years, the scaling of technology has promoted temperature to become one of the most important constraint in integrated circuit (IC) design. This is because the exponential growth in available logic resources has induced a rapid growth in chips power density which in a consequence has contributed to their operating temperatures. This puts strain on the static power dissipation as leakage-induced power consumption is exponentially coupled to the operating temperature, while increased leakage power increases temperature even further. Also, the reliability of ICs is affected by thermal gradients that cause mechanical stress of the silicon accelerating chip aging. Additionally, on-chip current densities have grown so high, that only a portion of a chip can be activated at a time. Those factors have necessitated taking a pro-active approach to the temperature management of computational devices.

Among chips that are typically implemented using the state-of-the-art nanometer technologies are high-performance FPGAs (Field Programmable Gate Arrays). They provide large amounts of configurable processing power in the form of a general purpose programmable fabric but also include many specialized high-performance blocks: DSP cores, memories, standard communication interfaces, even complete microprocessor cores. However their performance results in high power densities, therefore they suffer from thermal constraints. Since such devices can exhibit local and design dependent hotspots it is beneficial to be able to incorporate thermal sensors in a design to monitor its temperature on-line.

One way of predicting the thermal behavior of FPGA-based System-on-Chip (SoC) is with thermal simulation. However, most vendors only provide very simplistic Compact Thermal Models (CTMs) of their devices that adhere to the DELPHI methodology. Such models are suitable for system simulation but do not properly reflect thermal gradients within a die. On the other side of the spectrum – using accurate Finite Element Method (FEM) models is usually prohibitive in terms of labour and computation required. Consequently, there is a need for building CTMs of FPGAs that reflect thermal relations between different parts of SoC system, such as varying IP cores. However, in order to develop such models, scrupulous measurements are needed to formulate or validate thermal models of devices.

The contribution of this work is a novel method of physical emulation of heat transfer in ICs using physical FPGA-based heat transfer models. We describe our in-house built toolset and a flow designed to conduct both such emulations as well as detailed temperature measurements. As a by-product our method allows to monitor temperature of custom FPGA designs.

In order to achieve such an experimental environment, we manipulate low-level description of the FPGA-based systems in order to place temperature sensors and heaters where needed with fine-grained control over their physical layout. As a result, our toolset achieves: rapid preparation of experiments, high sensing resolution, and a low cost of individual sensors.

The ability to sense temperature of an FPGA device and to locally heat it up provides two significant benefits to the FPGA-based SOC designer: *i)* design testing and temperature monitoring – the designer can inject temperature sensors with the required communication infrastructure into his custom design and quickly evaluate its thermal behaviour; *ii)* evaluation of thermal models of the device – by placing a set of controllable heaters and temperature sensors, thermal response of a system to user-determined stimuli can be rapidly emulated and used to validate thermal models as was done in [1].

2 Related work

Operating temperature of a commercial-grade FPGA is typically limited to 125°C. However, operating an actual chip near this temperature accelerates the IC wear-out

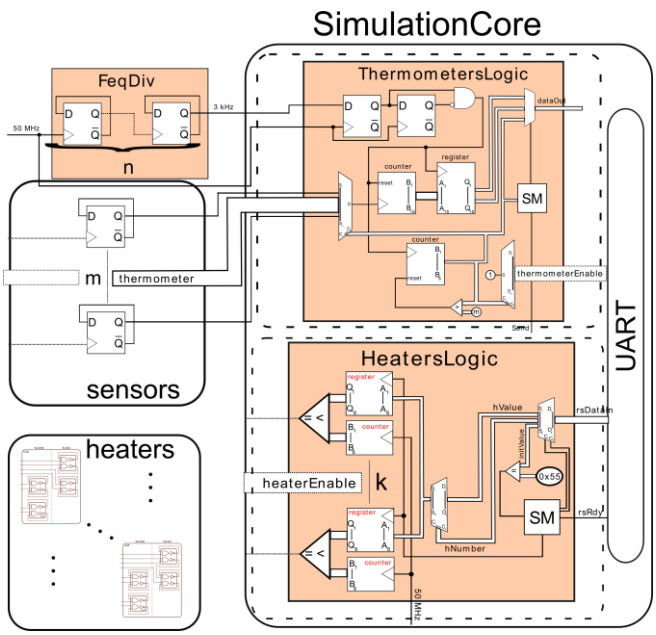

Figure 1: Overall architecture of the measurement system

caused by multiple temperature-induced ageing effects, such as electromigration or time dependent dielectric breakdown (TDDB). This justifies modeling the temperature-dependent reliability of programmable logic devices as well as online monitoring of the die's temperature.

While high-performance cooling systems can allow for the worst-case design, in embedded systems factors like unit cost, form factor and device volume may render them infeasible. Modern FPGAs exhibit high spatial thermal gradients, often exceeding 20°C [2]. These are caused by varying utilization of their configurable fabric, but also by the fact that contemporary high-end FPGA platforms incorporate multiple hard blocks such as dedicated processors, multipliers or DLLs, which increase differences in power densities across the die [2].

Temperature should be monitored and managed during operation of the system. For this high-end FPGA devices have built-in thermal sensors and vendors provide IP cores that allow to use them in custom designs. In particular designs, hotspots may occur far from the built-in sensor, which would suggest using multiple sensors. Even high-end devices are equipped with only a single temperature sensor. For example Xilinx Virtex devices starting from Virtex-5 family have one temperature sensor, available through the System Monitor [3].

Analog sensors used in integrated circuits produce voltage or current levels that are proportional to temperature of the sensor. An A/D converter is then used to obtain a useful reading. Digital sensors can be implemented using a ring oscillator (RO), a counter and a time reference. The system counts oscillations of an odd-numbered ring of inverters, then relates the result to some time reference. Due to near-linear dependence of logic delay on temperature, sensors built with ROs are very often used in reconfigurable logic [4], [5], [6], [7]. First works on sensing temperature in FPGAs using ROs [4], [8] demonstrated the concept and clearly listed advantages of such approach: i) hotspots and signal contentions can be detected if they manifest themselves with increased temperature, ii) no dedicated hardware is needed as sensors can be inserted and removed at the design time or even reconfigured dynamically, iii) the sensors are purely digital and need only operate for a very short time, hence with negligible contribution to both power consumption or the temperature rise itself.

Zick and Hayes [9] presented a sensor node implementable in reconfigurable logic which is smaller than the previous smallest designs while also being self-contained, i.e., requires no external logic. They also show procedures for inferring fine-grained variations in delay, temperature, switching-induced IR drop, and leakage-induced IR drop. All experiments were performed with over 100 sensors in a Virtex 5 device.

Thermal sensors planted across the chip require a controller unit responsible for scheduling and collecting measurements. The overhead in both timing and programmable resources related to the existence of that controller scales up with the amount of sensors in use. Consequently, with limited number of sensors, it is highly advisable to place sensors in places of interest – close to possible hot spots. That constraint poses a design problem to be solved. Mukherjee et al. [11] proposed algorithms for sensor placement in reconfigurable logic that helped reduce the number of sensors required to maintain high accuracy in monitoring hot spots. Sayed and Reda [11] proposed the use of soft sensors to circumvent the problem of sensor placement in reconfigurable devices, where exact positions of hot spots may change based on current configuration and workload. Sensor placement is constrained by the distance to a possible hotspot and the allowed error margin. An analytical model for upper bound of on-chip temperature differences was for example presented in [12].

3 Architecture of the measurement system

The overall architecture of our measurement system is depicted in Fig. 1. It consists of a set of temperature sensors, an optional set of heaters and control and communication logic implemented in an IP-core called SimulationCore. The SimulationCore is the only element of the whole toolset that has to be synthesized with the target design. In case when one wants to measure thermal response to an artificial set of power stimuli, a pre-synthesized version can be used to quickly start an experiment. The logic in the SimulationCore can support sequential readouts from up to 2^n sensors and control the power output of up to m heaters where n, m are limited by the amount of configurable fabric that we are willing to use. The logic controlling the thermometers consists of a frequency divider for providing the time-base, a demultiplexer for setting enable signal to each sensor,

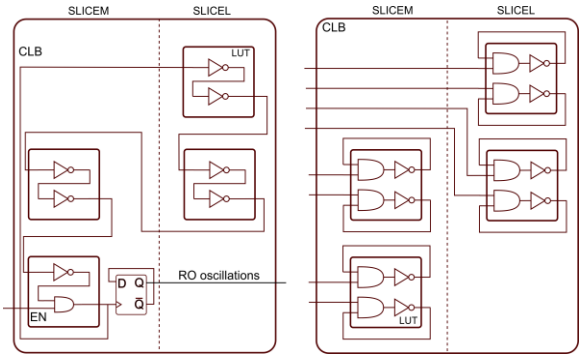

Figure 2: Left: schematic description of temperature sensing circuit implemented in one CLB. The sensor is a 7-stage ring oscillator. Right: schematic of a heater implemented in one CLB. The heater consists of eight 1-stage oscillators.

a counter and register for storing the RO oscillation numbers that is connected to all sensors via a multiplexer and a counter that controls this multiplexer and provides sensor number for the readout package.

The main part of the heaters logic part of the SimulationCore consists of an array of register and counter pairs for each heater. Each counter is clocked with the main 50 MHz clock. These pairs provide compact and efficient pulse-width modulation (PWM) implementation for each heater – it is enabled only when the counters' value is higher than what is in the heater's register. With 8-bit counters/registers 0.4% resolution is possible, but it can be adjusted easily.

Operation of the SimulationCore module is controlled with two compact Finite State Machines (FSM). One is responsible for setting the configuration registers for each heater. When a 3-byte set heater command comes (first byte equal 0x55), the second byte is used as the heater address and the third as it's setting. The other FSM is controlling the thermometers, multiplexing enable signals to the sensors, counting their oscillations in a predefined sensing time and also communicating with a computer via UART. Each readout is stored in a 3-byte package with 1 B sensor number and 2 B of counter value.

Temperature sensors were designed independently of the SimulationCore and can be easily changed. Using the centralized approach to gather the readouts we can have very compact sensors being just a RO with enable signal and an output for counting the oscillations. Each logical heater in our architecture can consist of any number of one-CLB heater modules. As with temperature sensors the heaters are controlled by the SimulationCore, but are designed independently and can be exchanged.

4 Temperature sensor design

RO-based temperature sensor measures temperature by counting the number of RO oscillations within a fixed time. As the oscillation frequency of the RO is temperature-dependent, the number of oscillations observed within the measurement window changes with sensor's temperature. The complete measurement setup is usually composed of three elements: a ring oscillator with an odd number of inverters and a single AND gate, acting as on/off switch (see Fig. 2), an oscillation counter, and a time-reference counter. A temperature sensor can be fully integrated with both counters, and able to operate stand-alone; or it can be distributed reducing the sensing element itself to just a RO, while the counters and the control unit can be dislocated elsewhere and/or shared by many or even all sensors. The latter architecture actually trades off the area/resource overhead of the complete measurement system with the number of measurements it can snapshot across the die simultaneously. This allows for a very compact implementation of the sensing element itself, at the expense of the system's temporal resolution. In our design, sensors share one common oscillation counter, therefore have to be read one by one.

An important trade-off in designing a RO-based sensor is between area and noise. The longer the inverter chain is, the more stable its oscillations are, the lower the noise, and the lower its power density. On the other hand, the size of the sensor grows with the length of the inverter chain. Typically authors use between 3 and 21 inverters in one chain (see Tab. I). To reduce internal fragmentation of resources RO chain lengths should be aligned with the capacity of a configurable logic block (CLB). Having this considered, we settled at a 7-stage RO that can be fitted in one CLB of our test device (Fig. 2).

In our design, we use a n-bit wide time-reference oscillation counter to measure the number of RO oscillations during a predefined time. At the beginning of the measurement, the AND gate, serving as an enable switch, is turned on and simultaneously a clock counter starts counting oscillations. Some authors [13], [14] advise to start the measurement after the RO stabilizes. However with sufficiently long measurement time this does not bring any improvement in terms of noise reduction.

Naturally, sensors generate heat when active. Thus, the measurement time has to be limited, such that the resultant amount of heat generated by the sensor and more importantly, its contribution to the local temperature of the die, remains negligible. In our setup, we were able to limit measurement time even down to 10 µs, however at the cost of substantial noise. Since sensors are enabled and disabled one after another, with high numbers of them their self-heating is negligible.

4.1 Sensor calibration

Temperature sensor implemented in FPGA device requires calibration. Due to process variation and nondeterministic routing algorithms this calibration should be repeated for each sensor in any given device. Three main approaches to this calibration process are described in literature. One is

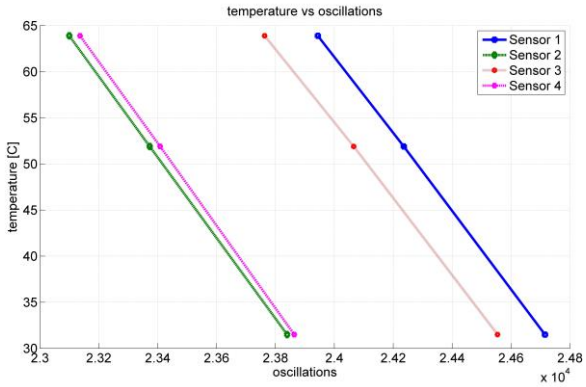

Figure 3: Regression results for the oscillations to temperature relation. Only 4 of 150 sensors were chosen for image readability – including the slowest and the fastest RO.

based on physical thermal sensor integrated in some devices [5], [15]. Another calibration method measures RO frequencies in a controlled environment such as a climate chamber. For each temperature level, the frequencies are logged and then linear or polynomial regression is performed against the data [7]. A similar method relies on IR measurements of the device. These two methods assume the temperature of all the sensors is equal to that of the package. This requires the device to be idle and dissipate minimal power uniformly over its area. In any case the calibration process should take into account the thermal properties of the device as well as the nature of the sensor.

We calibrated our sensors by measuring RO oscillations in idling device simultaneously with IR measurements of FPGA's package temperature in steady state. Apart from ambient conditions where package temperature was close to 32°C, we used an external heat source to obtain 2 more stable measurement points at 52 and 64°C. We found that the temperature-frequency relation is linear for the temperature range tested. There is a systematic variation in sensor frequencies that is common to both tested devices. In each device there is a hotter region in the upper right corner and cooler in the bottom left (Fig. 5, left). Unfortunately, this intra-sensor variation is quite high and it actually exceeds the dynamic range of our experiment. Differences between slowest and fastest ROs reach 6% when the device is in idle mode.

5 Heater design

Controllable heating sources have a number of applications in experimental research on thermal behavior of both FPGAs and ASICs. First, they allow to replace simulation with real experimentation in the evaluation, characterization, and calibration of heat flow models [5], [16]. Also, they enable accurate thermal characterization of the device itself, its power density as well as the impact of variability on its thermal response. Finally, controllable heat generation units were also used to improve and validate IR monitoring techniques and temperature-to-power mapping algorithms [17]. Unfortunately, authors of most publications do not describe their architectures, designs, or performance results of their heaters in much detail. The only systematic study of the design of a programmable heating elements in configurable logic is [18] where eight different heat generating cores, implemented using different resources available in the FPGA matrix, are evaluated in terms of their maximal temperature.

In our work we use controllable heaters to evaluate and characterize our temperature sensing infrastructure. Similarly to [18], we base our heaters on length-of-1 ring oscillators which provide very high frequencies and substantial power density. In our implementation, each such singular heater occupies one CLB (see Fig. 2). Those CLB-sized heaters can be banked together in order to operate as one larger heater. The advantage of having controllable heaters implemented as single CLBs is that they can be placed anywhere on the FPGA fabric as long as routing resources allow to route them together.

We control heaters output power by PWM. Each heater bank enable line is controlled by a dedicated counter, a control register and a comparator that all together provide the PWM on the heater bank enable line, with the default power resolution of ca. 4%. Due to high frequency of on-off transitions, that is much faster than thermal phenomena that we observe, the heaters provide good control over power output with a linear characteristic.

In our implementation device, there are 4 slices in a CLB, each having a Look Up Table (LUT) which can implement any function of 4 inputs. This is enough to implement 8 heaters with the length-of-1 ROs, yielding the highest power density. To verify the heaters' parameters, we measured their power consumption by current measurement at a jumper that was specifically designed for that purpose by the board vendor. Our first estimates of power and power density of one-CLB heater yield at least 0.3 W/mm^2 (the measured power is 6 mW per one-CLB heater; the die is at most 10×10mm in size and holds a reprogrammable matrix of 65×88 CLBs).

Unfortunately, with high total heat size and output setting, the load-induced IR drop causes the heaters to not behave as expected. With total output higher than 3.5 W heaters start to slow down, yielding diminishing increase with rise of heater size. We expect that the underlying cause of this behavior is that the ROs are sensitive to load-induced voltage variations [11]. Hence, in future work we will substitute the current asynchronous heaters with a synchronous design.

6 Toolset

Our toolset is composed of four components:

1. **FloorplanMaker**. A GUI tool for inspecting the floorplan of a given project and specifying the placement of temperature sensors and heaters

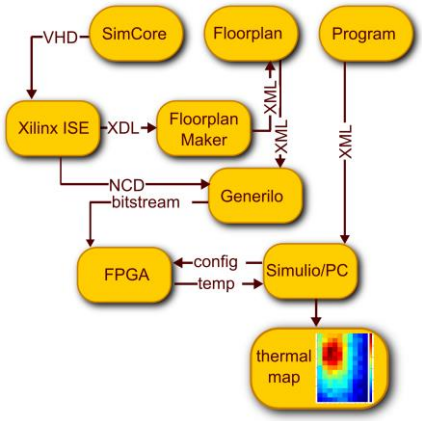

Figure 4: Toolset flow chart

2. **Generilo**. A command line tool that inserts temperature sensors and heaters into the design by manipulations of XDL files using the API described in [19];
3. **SimulationCore**. An IP core for communication with an external computer and controlling the temperature sensors and heaters;
4. **Simulio**. A command line tool for communication with the SimulationCore.

Standard vendor's synthesis flow for FPGA consists of a few main steps. From the system description in HDL files (*.vhd) a synthesis process produces a Native Generic Database in the form of *.ngd files. Then, Map and Place & Route stages produce an *.ncd file with physical description of FPGA components used in the project and connections between them. The final stage is the bitstream generation which produces *.bit file suitable for programming the device based on the *.ncd file.

To be able to add temperature sensors to an existing design two main steps must be performed. The SimulationCore must be fitted into the design and made available at the highest level of HDL hierarchy and a *.ncd file from the synthesis must be made available. Then, the toolset flow is as follows (see Fig. 4):

1. synthesize the design with SimulationCore configured with desired number of sensor and heater support,
2. load the XDL file into FloorplanMaker, add temperature sensors in desired positions,
3. save the positions of the sensors using GUI,
4. run Generilo to add sensors to the design and automatically run vendor's tools to generate the bitstream,
5. program the device and communicate with the SimulationCore via serial port using Simulio.

Using all the tools is very straightforward, especially adding temperature sensors using the Graphical User Interface (GUI) of the FloorplanMaker. After importing the current design from a *.ncd file, the reserved blocks of the device are presented in dark grey, and current design in light gray. Then, using mouse clicks, sensors and heaters can be added to the design and saved in an XML file. Using XML files makes it convenient to store and modify experiment's configuration. It also allows to use scripts to e.g. generate a dense grid of sensors and later-on load it into the FloorplanMaker for visual inspection.

As can be seen in Tab. 1 even using up to 256 thermometers requires modest area to accommodate the SimulationCore IP. One temperature sensor occupies only one CLB, which allows to place more than 211 sensors with the required control logic in the Spartan 3E 1600 device Also the Spartan 3E is not a very large FPGA in terms of the amount of available logic. Hence, with newer devices resource utilization should not be a concern.

7 Experimental evaluation

Currently our main development platform is the Spartan 3E-1600 development board from Agilent. The Spartan 3E-1600 from Xilinx provides large amount of generic configurable logic and little specialized blocks which is convenient for placing sensors and heaters. Its 320-pin FBGA package has a relatively high junction-to-ambient thermal resistance (Θ_{JA}) of 21.1°/W. The board itself supports FPGA current measurements on a dedicated jumper. This allows to measure both supply voltage and current flowing through the device.

We synthesized the Simulation core with varying number of logic supporting 16–256 temperature sensors and 8–64 heaters. The device utilization results are presented in Tab. I. We show the number of flip-flops and LUTs used as well as percentage of total utilization of device configurable fabric. It is worth highlighting that using up to 128 sensors requires only 4.7% of available resources.

Table 1: Configurable resources utilization by heating and sensing infrastructure. "Control only" means only control logic on-board, "all" means with thermometers and heaters on-board. Presented values are post-map, area optimized, results obtained using Xilinx ISE 13.3 (lin64). xT + yH – x thermometers and y heaters in the design.

	Spartan 3E architecture (device – xc3s1600e)			
	Slice FFs	**4-Input LUTs**	**% of device (control only)**	**% of device (all)**
state machine	53	33	0.18	-
32T	168	116	0.57	1.44
64T	235	183	0.78	2.52
128T	362	276	1.23	4.70
256T	623	496	2.12	9.06
16T + 4H	269	171	0.91	1.45
32T + 8H	399	257	1.35	2.44
64T + 16H	659	444	2.23	4.39
128T + 32H	1171	766	3.97	8.31
256T + 64H	2201	1423	7.46	16.13

To test our toolset we placed a 15×10 sensor grid in the device. Then we added a heater spanning 15_16 CLBs, with 9 sensors in it and tested the thermal response of the device.

Using sensor calibration coefficients from section IV we measured thermal response of the device. We achieved increase in temperature of 10–26 K, relative to the idle device. The resulting thermal map can be seen in the right-hand part of Fig. 5.

The SimulationCore communicates with a PC via UART and RS-232/USB converter. We found the communication speed of the SimulationCore and serial interface sufficient with updates on up to 256 thermal sensors every 84 ms on average with a standard deviation of 18 ms. The maximal time between two consecutive temperature readings reached the computer was 110 ms. This can be improved by reducing either the available number of sensors or reducing sensing time. However, the latter is not advisable since it will lead to increase in the amount of sensor noise.

8 Conclusions

We present a complete, working toolset ready to conduct thermal experiments with Spartan 3E-series devices. Using this toolset one is able to insert a large set of temperature sensors into an FPGA-based design, and do post- synthesis. The architecture of the system is highly configurable, but using the default settings and a set of tools with graphical user interface and clear XML-based configuration. System allows for rapid experiment setup and provides high flexibility with very little overhead in terms of logic usage.

In the future work we intend to analyze a wide variety of tradeoffs associated with architecture of the system: sensor noise, temporal resolution and control logic overhead. We are also currently working towards porting our toolset to Spartan 6 and Virtex 5/6 device families to provide support for newer technology.

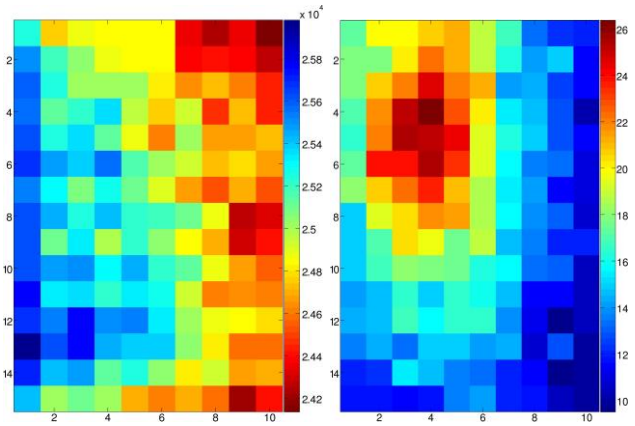

Figure 5: Left: Frequency map of sensors in an idling device. Higher oscillation count is related to lower temperature. Right: Map of temperature rise with one heater dissipating power relative to idle mode of Spartan 3E1600.

Acknowledgements

This work was supported by National Science Centre grant no. 2011/03/B/ST6/00343.

Literature

[1] S. Velusamy, W. Huang, J. Lach, M. Stan, and K. Skadron, “Monitoring temperature in FPGA based SoCs,”, ICCD 2005, IEEE Int. Conf., Oct. 2005, pp. 634 – 637.

[2] Sundararajan, A. Gayasen, N. Vijaykrishnan, and T. Tuan, “Thermal Characterization and Optimization in Platform FPGAs,” in ICCAD ’06., nov. 2006, pp. 443–447.

[3] Virtex-5 FPGA System Monitor User Guide, Ug192 v1.7.1 ed., Xilinx, Feb. 2011.

[4] E. Boemo and S. L opez-Buedo, “Thermal monitoring on FPGAs using ring-oscillators,” in Field-Programmable Logic and Applications. Springer, 1997, pp. 69–78.

[5] M. Happe, A. Agne, and C. Plessl, “Measuring and Predicting Tem¬perature Distributions on FPGAs at Run-Time,” in ReConFig 2011, Int. Conf., 30 2011-dec. 2 2011, pp. 55–60.

[6] S. Lopez-Buedo and E. Boemo, “Making visible the thermal behaviour of embedded microprocessors on FPGAs: a progress report,” in Proc. FPGA ’04., NY, USA: ACM, 2004, pp. 79–86.

[7] K. M. Zick and J. P. Hayes, “On-line sensing for healthier FPGA systems,” in Proc. FPGA ’10. NY, USA: ACM, 2010, pp. 239–248.

[8] S. Lopez-Buedo and E. Boemo, “A method for temperature measurement on reconfigurable systems,” in Design of Circuit and Integrated Systems Conf. Citeseer, 1997, pp. 727–730.

[9] K. M. Zick and J. P. Hayes, “Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems,” ACM Trans. Reconf. Technol. Syst., vol. 5, no. 1, pp. 1:1–1:26, Mar. 2012.

[10] R. Mukherjee, S. Mondal, and S. Memik, “Thermal sensor allocation and placement for reconfigurable systems,” ICCAD ’06. IEEE/ACM International Conference on, Nov. 2006, pp. 437–442.

[11] M. Sayed and P. Jones, “Characterizing Non-ideal Impacts of Reconfigurable Hardware Workloads on Ring Oscillator-Based Thermometers,” ReConFig, 2011 International Conference on, Dec. 2011, pp. 92–98.

[12] S. Sharifi and T. Rosing, “Accurate Direct and Indirect On-Chip Temperature Sensing for Efficient Dynamic Thermal Management,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 29, no. 10, pp. 1586–1599, Oct. 2010.

[13] S. Lopez-Buedo, J. Garrido, and E. Boemo, “Dynamically inserting, op¬erating, and eliminating thermal sensors of FPGA-based systems,” Components and Packaging Technologies, IEEE Transactions on, vol. 25, no. 4, pp. 561–566, Dec 2002.

[14] R. Mukherjee, S. Mondal, and S. Memik, “A sensor distribution algorithm for FPGAs with minimal dynamic reconfiguration overhead,” in In Proc. IEEE Int. Conf. on Engineering of Reconfigurable Systems and Algorithms, June 2006.

[15] C. Ruething, A. Agne, M. Happe, and C. Plessl, “Exploration of ring oscillator design space for temperature measurements on FPGAs,” FPL 2012, 22nd Int. Conf., Aug. 2012, pp. 559–562.

[16] J. Franco, E. Boemo, E. Castillo, and L. Parrilla, “Ring oscillators as thermal sensors in FPGAs: Experiments in low voltage,” in SPL 2010, March 2010, pp. 133–137.

[17] S. Reda and A. N. Nowroz, “Power Modeling and Characterization of Computing Devices: A Survey,” Found. Trends Electron. Des. Autom., vol. 6, no. 2, pp. 121–216, Feb. 2012.

[18] M. Happe, H. Hangmann, A. Agne, and C. Plessl, “Eight ways to put your FPGA on fire – A systematic study of heat generators,” ReConFig 2012 , 2012, pp. 1–6.

[19] K. Kepa, F. Morgan, K. Kosciuszkiewicz, L. Braun, M. H¨ubner, and J. Becker,, “Design assurance strategy and toolset for partially reconfigurable fpga systems,” ACM Trans. Reconfigurable Technol. Syst., vol. 4, no. 1, pp. 4:1–4:26, Dec. 2010.