A New Temperature Distribution Measurement Method on GPU Architectures Using Thermocouples †

Aniruddha Dasgupta[1], Sunpyo Hong[1], Hyesoon Kim[2] and Jinil Park[3*]

[1] *School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA*
[2]*School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA*
[3]*School of Mechanical Engineering, Ajou University, Suwon, 443-749, Korea*

Abstract

In recent years, the many-core architecture has seen a rapid increase in the number of on-chip cores with a much slower increase in die area. This has led to very high power densities in the chip. Hence, in addition to power, temperature has become a first-order design constraint for high-performance architectures. However, measuring temperature is very limited to on-chip temperature sensors, which might not always be available to researchers.

In this paper, we propose a new temperature-measurement system using thermocouples for many-core GPU architectures and devise a new method to control GPU scheduling. This system gives us a temperature distribution heatmap of the chip. In addition to monitoring temperature distribution, our system also does run-time power consumption monitoring. The results show that there is a strong co-relation between the on-chip heatmap patterns and power consumption. Furthermore, we provide actual experimental results that show the relationship between TPC utilizations and their active locations that reduce temperature and power consumption.

Keywords: Temperature Measurement; Many-core Architecture; GPU; Thermocouple

1. Introduction

The number of cores inside a chip is increasing dramatically in today's processors. For example, NVIDIA's GTX280 has 30 streaming multiprocessors with 240 CUDA cores, and NVIDIA Fermi GPUs have 512 CUDA cores. On the multicore front, the latest AMD processors have 12 cores. This high number of cores puts a lot of pressure on designing effective power- and temperature-controlled architectures. Moreover, the work by Mesa-Martinez et al. [10] showed that temperature is becoming a dominant factor for the determining performance, reliability, and leakage power consumption of modern processors.

In this paper, we use GPUs as a form of many-core processor. With GPUs, it is possible to validate that temperature-aware thread scheduling can actually reduce power consumption. Unfortunately, unlike the state-of-the-art multicores, the current GPUs do not provide temperature sensors for each individual core. Usually, a board-level temperature sensor is provided. However, it cannot account for the rampant temperature variations across the chip due to hotspots. Hence, in this paper, we propose a new temperature-measurement system that allows us to measure the temperature map, while also measuring the total power consumption.

Some efforts in academia have focused on measuring temperature using infrared (IR) cameras [11] (Although industries have better ways of measuring temperature, typically that information is not disclosed to the public). IR cameras provide an entire temperature distribution, but set-up cost is very high, and they require special oil cooling. In other words, a heatsink must be removed, which could interfere with the natural heat distribution from a heatsink. Also, measurements performed through such a setup typically require some adjustments to the measured data, so as to accurately represent ideal measurements under actual working conditions of the processor (i.e., with a heatsink cooling solution). Thus, there is an opportunity for inaccuracies to creep in due to the nature of the modeling.

Hence, we propose a new cost-effective temperature-measurement system that uses thermocouples for the first time for GPU architectures. We devised a method to install thermocouples between a chip and a heatsink. With this system, we successfully measured the on-chip temperature distribution of a GPU processor. Thermocouples provide two benefits over IR cameras. First, they are very low cost and relatively easy to install, even in academia, without special expensive equipment. Second, a heatsink can still be placed, so we can measure power and temperature simultaneously. Then, we demonstrate the need for thermal-aware scheduling algorithms based on the correlation between the on-chip heatmap and power consumption.

2. Background and Related Work

In this section, we discuss previous chip temperature-measurement systems and provide a brief background of the evaluated GPU system.

2.1 Chip Temperature Characterization Methods

Chip temperature characterization methods can be classified into two main branches: 1) modeling methods, and 2) measurement methods.

2.1.1 Modeling Methods

Temperature-modeling methods are mainly relevant to design time thermal characterization. They provide designers with the freedom to try out new designs and perform simulations. Also, such thermal models can be plugged into microarchitecture simulators to see the effect of changing microarchitectural parameters on temperature or the effect of running different benchmarks. One of the most popular thermal models is HotSpot [9]. Based on the duality of heat transfer and electricity, the authors have modeled various microarchitecture components into equivalent thermal resistances and capacitances. HotSpot can also be used to model a particular thermal package for the chip and to observe its thermal characteristics. By plugging the HotSpot thermal model into a simulator, one can track the thermal properties of individual components under load, understand a program's thermal behavior, evaluate thermal management techniques, etc. The thermal model is portable and flexible, and it can be built upon to cater to particular requirements. However, verification of the model on absolute thermal values is still a challenge.

2.1.2 Measuring Methods

Temperature measurement methods are mainly relevant to runtime thermal management techniques, which require a temperature measurement to occur in real time. Also, though thermal simulation models aim to faithfully mirror the behavior of the system, they are based on the designer's understanding of what factors affect the thermal characteristics of the system. So, modeling methods need to be validated against actual measurements of some sort to ensure the accuracy of the model and thus they require the existence of robust thermal-measurement methods. In the realm of performance, modern processors provide measurement instruments in the form of hardware performance counters. However, for temperature, processors, especially many-core processors, do not yet have a concrete built-in measurement system. Though the exact methods used in the industry to measure temperature are not known, there are mainly two contemporary methods proposed in academia.

On-Chip Sensors: CMOS-based on-chip sensors are mainly used to measure temperature at various points. This type of temperature sensing has been well-implemented in multi-core processors, with each core having its own thermal sensor. The IBM Power6 processor has 24 digital thermal sensors and three thermistors for monitoring temperature characteristics [5]. But in the case of many-core architectures like GPUs, so far there is just a board-level sensor [3] and one on-chip sensor whose location is unknown; the temperature of individual cores is not tracked. The advantage of using on-chip sensors is the accurate and real-time measurement of temperature across the chip, without the need for alternate cooling solutions, as in the case of IR. Thus, temperature monitoring can be performed in the actual working conditions of the chip running real workloads. This translates to more accurate handling of DTS techniques. On the downside, some problems exist due to the sensors being integrated into the chip. Due to variations in the lithographic process, a complicated sensor circuit is required to achieve accurate results. This establishes a tradeoff between accuracy and the amount of die area taken up by the sensor circuitry. Also, since sensor locations are discrete in nature, sensing all the hotspots on the chip is not possible, which leads to a spatial gradient of error if a sensor is not at the exact location of the hotspot.

IR-based Measurement: Infrared-based thermal imaging has gained popularity as a robust method of characterizing thermal behavior [11]. It provides good resolution and accuracy both in time and space. As such, it has been used in studying dynamic thermal management techniques. Its external nature also helps in making decisions regarding the placement location of thermal sensors on the chip at temperature-critical portions. However, there are a few limitations of using IR imaging, some of which have already been pointed out by Huang et al. [7]; IR rays cannot pass through metal. Generally, processors are encompassed with a metallic heat dissipation solution like a heatsink. So, for IR imaging to work, the heatsink needs to be removed and an alternate cooling solution needs to be provided. One of the prevalent methods in this case is removing the heatsink and providing laminar oil-cooling over a bare silicon die [11]. However, this results in different transient and steady-state thermal responses compared to a conventional cooling solution like a heatsink [7]. The other limitation is that the cooling capacity of oil is roughly proportional to the size of the oil tank and the velocity of the oil flow. In order to cool 100W-300W cores, the speed of oil flow has to be fast, thereby easily producing more distorted images. Although this method has high merits when done correctly, it comes with high set-up cost and time.

2.1.2 Pros and Cons of using Thermocouples

The most notable advantage of using thermocouples is the cost and the ease of use for the measurement. Not only is it suitable for measurements up to 750 degrees Celsius, but it has a very thin diameter, and being a wire, it can easily be placed anywhere. However, placing this wire in a specific location is a challenge as the pressure applied on it could affect the temperature readings. Furthermore, the resolution of the readings is very limited to IR measurement. However, with a very well designed thermo-spacer and some knowledge of the GPU processor layout, using thermocouples provides the best cost-effective solution. Also, reconstructing a heatmap from thermocouple readings is much simpler than in the IR case since the IR method involves high-velocity oil flow. To the best of our knowledge, actual temperature measurement on a GPU chip has not been done before, and unlike the CPU architecture, GPU has a very high number of cores and has more opportunities for temperature and power reduction from this study.

3. Experimental Setup

Figure 2 shows a block diagram of the entire temperature- and power-measurement system. The AC power is intercepted by the EXTECH power analyzer, which then is connected to the test computer. The computer has an 8800GT GPU with thermocouples and the spacer installed. Thermocouple readings are measured by another computer using Labview software.

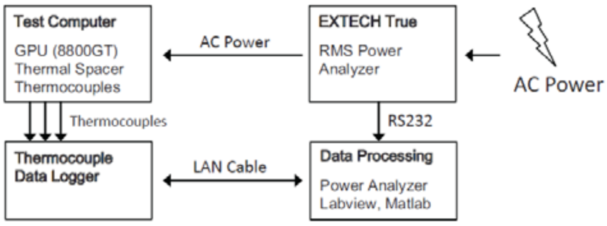


Fig. 1. Temperature- and power-measurement system.

3.1 Temperature Measurement System

We propose a thermal-measurement method where thermocouples are used as temperature sensors. We have designed a thermal spacer with grooves cut in to hold the thermocouples at desired locations. The thermocouples are embedded in these grooves. The spacer has raised edges and a shape such that it fits perfectly over the GPU chip, consequently establishing a contact between the thermocouples and the chip surface.

Thermocouple: J-type thermocouples are used in our measurement system. They are suitable for measurements ranging from 0 to 750 degrees Celsius, which is more than enough to cover the spectrum of temperatures encountered in a working GPU chip. They have a high sensitivity of around 55 uV/degrees Celsius. The J-type is one of the most popular thermocouple types because of its wide measurement range and superior voltage output, which translates to greater temperature resolution.

Thermal Spacer: The thermal spacer is made of copper, the same material as the heatsink on the GPU. Consequently, it transfers heat from the GPU to the heatsink very well. The thermal resistance of the spacer is so low that it can be ignored for all practical purposes. Thus, our temperature-measurement methodology does not affect the working of the GPU in any detrimental way.

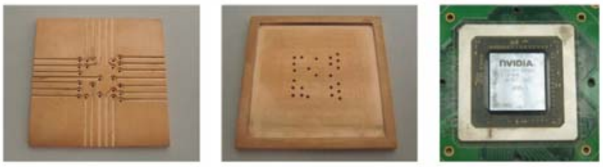


Fig. 2. Customized thermospacer for 8800GT GPU.

Installation-Methods Previously Attempted: Taping using heat transfer tapes, soldering, and gluing using thermo-epoxy are other possible installation options, but we learned that they are not feasible. Soldering does not work because the surface of a chip cannot be soldered. Both taping and gluing allow installation of thermocouples, but they have two serious problems. First, both tape and glue material themselves prevent heat transfer from the chip to the heatsink. Even with material specifically designed for high temperature, it is still not good enough to transfer all the heat from the chip. The second problem is that placing thermocouples exactly at the desired locations is not a trivial task.

Therefore, we used grooves in the thermal spacer to hold the thermocouples in place. The sensor placement pattern is uniform in nature so as to take temperature measurements on the GPU chip over a uniform pattern grid. A layer of thermal paste is applied on the GPU chip as well as on the thermal spacer to ensure smooth thermal contact throughout.

Figure 3 shows the thermal spacer, the locations of the thermocouples, a picture after the thermocouples are placed, and an estimated floor plan of the chip. The inner box indicates the actual chip size and the outside box is the size of the heatsink. Figure 3(a) shows an estimated floor plan of SMs. This floor plan is estimated based on GTX280 [4], which has the same microarchitecture but a different number of SMs. The floor plan shows the location of cores and TPCs. We estimate the core locations based on our one-core active experiments in Section 5.1.2. Figure 3(d) shows a side view of the installed thermocouples and the spacer between the heatsink and the chip.

Data Logger: The thermocouples are connected to a data-logger unit NI FP-TC 120, three 8-channel thermocouple modules for Field- Point [2]. We use a 10/100 MBps Ethernet interface for FieldPoint to communicate the sensor data to the data-logging machine.

3.2 Power Measurement System

We use the Extech 380801 AC/DC Power Analyzer [1] to measure the overall system power consumption. The raw power data is sent to a data-log machine every 0.5 seconds through an RS232 interface. Note that multiple computers are involved in recording power and thermocouple readings, so timing is synchronized.

3.3 Reconstructing Images

To reconstruct temperature images, Matlab is used. We have written a script that considers each thermocouple channel in a correct spatial location. Furthermore, we can simulate a specific slice of time as thermocouple data has been cumulated over a period of time. To interpolate between each thermocouple reading, a contour function is used to reconstruct an overall thermal image.

4. Many-core Architecture

Figure 4 shows the high-level view of a heavily multithreaded and many-core GPU architecture (NVidia's 8800GT is used). A series of streaming multiprocessors (SM) are connected by an interconnection network and to a DRAM system.

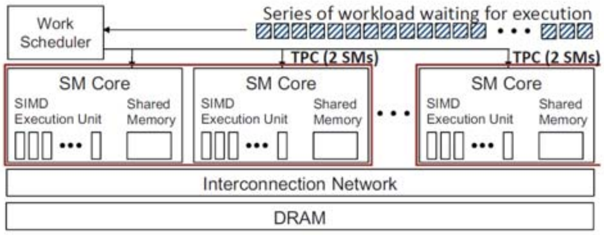


Fig. 4. High-level GPU architecture and workload execution.

The top of the figure shows a series of workloads that get scheduled by work scheduler unit. Unlike in the CPU architecture, scheduling is done purely by hardware. As a result, the number of activated SMs and which workload gets assigned to which SM are *non-deterministic*. This lack of understanding is a potential problem for this study, as we need to know what cores to turn on and keep them running without interruption.

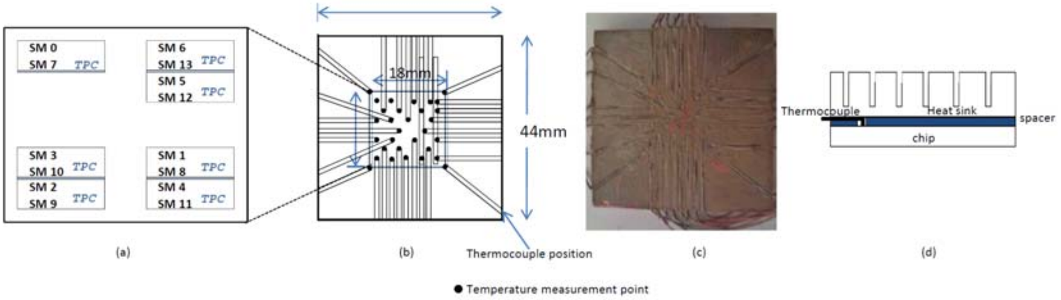
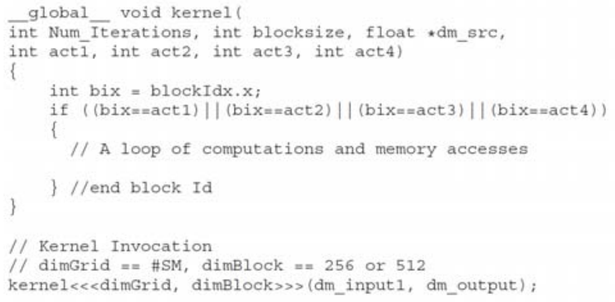


Figure 3: Temperature measurement system design (a) estimated floor plan of the GPU, (b) thermal spacer design (c) picture after thermocouples are placed on the spacer (d) a side view of the installed thermocouple and spacer.

4.1 How to control which core for execution?

Currently, GPU vendors do not disclose information on how to control the scheduling and other essential information. Hence, to overcome this problem, we devise a new technique in software to make sure that only a single workload gets assigned to each SM. Each workload is intentionally modified to use just the right amount of SM resources (i.e., increasing shared memory and register usage), so that only one workload gets assigned to an SM. Then, we intentionally invoke a number of workloads that is identical to the number of SMs in the GPU. Another modification is that we made each workload run for a sufficiently long time, as we do not want frequent context switching between workloads. For verification, when we increased by just one more workload, the execution time is doubled, which shows that all SMs were activated just before the workload addition. Figure 5 shows that by carefully modifying the *act* value, we can control which active core is used for execution. Note that not all real GPU benchmarks are constructed in this manner, and currently controlling a specific core with this technique using those benchmarks is not possible.



```
__global__ void kernel(
int Num_Iterations, int blocksize, float *dm_src,
int act1, int act2, int act3, int act4)
{
    int bix = blockIdx.x;
    if ((bix==act1)||(bix==act2)||(bix==act3)||(bix==act4))
    {
      // A loop of computations and memory accesses

    } //end block Id
}

// Kernel Invocation
// dimGrid == #SM, dimBlock == 256 or 512
kernel<<<dimGrid, dimBlock>>>(dm_input1, dm_output);
```

Fig. 5. Simplified view of code example.

The high-level view of this specialized benchmark has a number of floating point multiply-adds and coalesced memory loads inside a loop. We supply as parameters to the kernel all the SM numbers that should be active for the run. Figure 5 shows an example of activating four blocks. The benchmark is run for a fixed amount of time (120 seconds in the above case2) during which, the host code calls the kernel in a loop till the specified time is elapsed. We use the *nvclock* utility to record the GPU board temperature. Based on the benchmark output and the *nvclock* utility output, we calculate a running average of GPU board temperatures and also note the maximum temperature for each configuration run.

5. Results

5.1 Temperature Measurement System

5.1.1 Calibration Experiments

We design a calibration experiment system as shown in Figure 6. Two plates have been designed and manufactured, as shown in Figure 6. Plate 1 mimics the thermal behavior of a processor (heat source), and Plate 2 mimics the thermal behavior of a heatsink. One side of Plate 1 has the exact same shape of the chip, so we can place the spacer on which the thermocouples are already installed between two plates. We uniformly increase the temperature of Plate 1. After that, we place Plate 1 in the ambient temperature and install the spacer and Plate 2 in order. Then, Plates 1, 2, and the spacer reach the steady state, which is at room temperature. Figure 6 shows the calibration result, which shows that during the transient period, temperature differences occur, especially in the initial stage. We believe that these initial differences are primarily due to different physical pressures applied to some thermocouples when putting Plate 2 on top of Plate 1 physically. Once the weight of Plate 2 is stabilized on Plate 1, only minor temperature differences exist, especially in the calibration range (operation range). Hence, this shows that we can use thermocouples to measure the heat distribution on the surface of a processor.

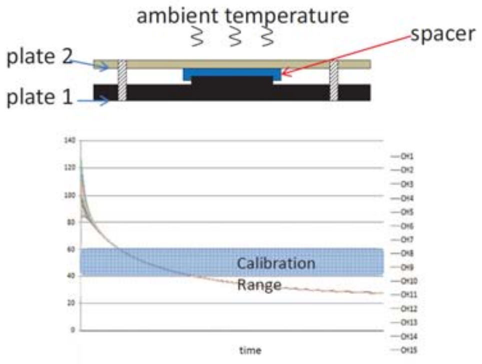


Fig. 6. Thermocouple calibration system (top) and the results (bottom).

5.1.2 One-core Activation

One of the important questions is whether there will be enough of a temperature difference between active cores and idle cores. To answer this question, we activate only one core at a time and vary the active core locations. We adjust the time of execution such that the temperatures reflected by the thermocouples reach a saturated value. We take an average of 30 readings after saturation for each thermocouple location and plot the heatmap at the saturation point taking this value. Figure 7 shows the heatmap of two different active cores (Core 1 and Core 4) and the idle state. The results show that when a core is active, the temperature is higher than in other areas by around 5 degrees. Please note that, even though the rest of the cores are idle, because there is no power gating or clock gating, those cores are still on, consuming some power.

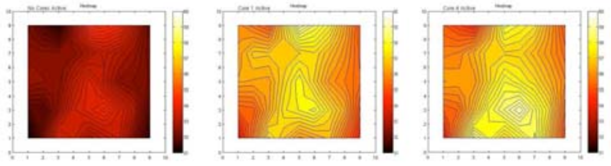


Fig. 7. Difference in heatmaps for idle and active cores (Left: no active core, middle: Core 1 active, right: Core 4 active).

On performing the one-core activation experiment, we observed that the heatmaps for 0 and 7 were very similar. This was also true for SM combinations of (1,8), (2,9), (3,10), (4,11), (5,12), and (6,13). So, it is apparent that 0 and 7 belong to the same TPC, and the same can be said about the other combinations. Figure 8 shows the similarity in thermal maps for combinations (0,7) and (3,10). Note that neither a default GPU scheduling algorithm nor exact core locations are disclosed by GPU vendors.

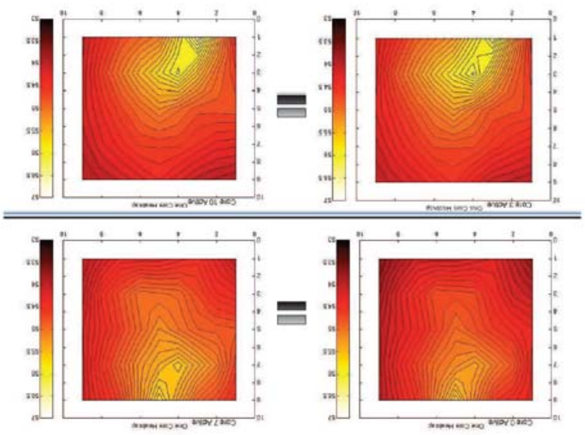


Fig. 8. SMs belonging to the same TPC (top left: Core 0 active, top right: Core 7 active, bottom left: Core 3 active, bottom right: Core 10 active)

5.1.3 Repeatability and Rotation Test

To test the stability of the thermocouple measurements, we performed a rotation test (the chip is isotropic). In this experiment, we insert the spacer after rotating it 90 degrees from the original position. If the temperature deltas that we observed in the original position were caused by the thermocouples themselves instead of the actual hotspot of the GPU, when we rotate the spacer, the hotspots would have rotated together. Please note that the thermocouples are already glued in the spacer, so when we rotate the spacer, the thermocouples are also rotated together. The default configuration is called 0 degree, and we plotted the heatmap with the spacer at 90 degrees. Figure 9 shows the results of the 0- and 90-degree experiments (the 90-degree data is also drawn based on the core locations in the 0-degree data). The results show that the heatmap at 90 degrees is very similar to 0-degrees heatmap, so the hotspot is still found correctly. Hence, we can say that the thermocouples are laid out properly to detect hotspots irrespective of the orientation. Although we do not present the results in this paper, we also did the repeatability test. We rotated the spacer back to the original position and compared the results with the initial the 0-degree experiment data. The repeatability test shows very similar results. These experiments point to the robustness of the temperature-measuring method using thermocouples with a custom-designed thermal spacer.

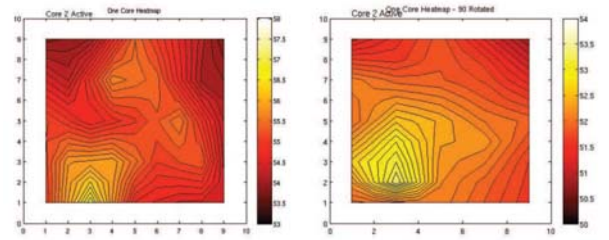


Fig. 9. The 0- and 90-degree heatmaps of thermal spacer with Core 2 active

5.2 Temperature and Power

5.2.1 Temperature Aware Scheduling

To save energy, many temperature-aware thread-scheduling algorithms have been proposed. The advantage of certain core combinations being thermally optimal or generating lower power can be explained by thinking about the layout from a thermal perspective. As explained in detail [8], interleaving high power density elements with lower power ones leads to virtual lateral heatsinks. Thus, when scheduling work on cores that are distant from high power density elements, scheduling such that active cores are separated by low power/cooler running components would give such a combination of active cores an edge from the thermal and power point of view. Thus, having an idea about the layout, one can intelligently schedule work to minimize thermal stress and power consumption. Also, a more uniform power and thermal distribution leads to lower hotspot formation.

5.2.2 Temperature and Power Measurement

Using our power-measurement system and results of the on-chip sensor, we can find the delta in power as well as temperature for different combinations of active cores.

We measure temperature and power together by activating one, two, four, and seven cores. For one-core and two-core tests, power consumption is almost the same regardless of which core(s) is(are) active. This is because one or two cores do not generate enough power to create severe hotspots. The seven-core test also shows similar power consumption behavior. This is because more than half of the chip is activated so the entire chip becomes hot (i.e., no temperature distributions.) We observe that activating four cores provides a significant delta, depending on core positions. Hence, we report the results of the four-core test.

5.2.3 Multiple-Core Tests on 8800GT

We tried different combinations of four-active cores in 8800GT and measured the power and temperature for each case. Table 1 summarizes the results, which show a strong correlation between temperature and power. Higher temperature consumes more power. From the table we can see that the core combination of 0-7-1-8 consumes the least amount of power and produced the lowest temperature, while the combination 0-1-2-3 induces maximum thermal stress and power. This is a very interesting phenomenon, as we are executing the same code on the same number of SMs (processors). This fact can be corroborated by looking at the heatmaps for the two cases shown in Figure 10. For the 0-7-1-8 case, heat is well spread, so the overall temperature is lower. However, for 0-1-2-3, the heat is concentrated in the center, so the overall temperature becomes higher. This is consistent with the average temperature measurement from the on-chip sensor. Furthermore, these results can be used to construct a predicted floor plan, which is shown in Figure 3(a). It is apparent that the 0-1-2-3 case activates four TPCs, while the 0-7-1-8 case activates only two TPCs.

Based on these results, we can conclude that temperature-aware thread/core scheduling can actually change the power consumption. When few cores are active, depending on which cores are hot, the overall temperature can vary much and so does the power consumption.

Active Cores	Avg. Power (Watts)	Avg. Temp (Celsius)	#Active TPCs (Estimated)
0-7-1-8	253.68	76.99	2
4-11-6-13	253.77	76.59	2
2-9-5-12	254.44	77.42	2
4-11-0-1	256.36	77.44	3
3-10-5-6	257.23	78.01	3
6-7-8-9	261.04	79.47	4
10-11-12-13	261.66	78.88	4
2-3-4-5	261.77	79.60	4
0-1-2-3	262.53	80.41	4

Table 1: Four active cores – measured power vs. on-chip sensor.

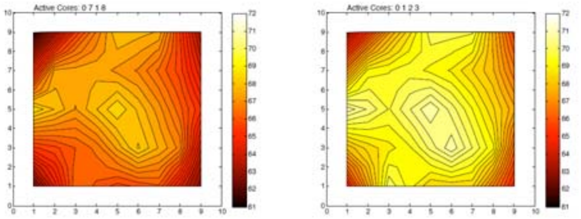


Fig.10. Difference in heatmaps between high and low thermal stress (left: 0-7-1-8, right: 0-1-2-3).

5.2.4 Projection of Thermal Effect on Higher Number of Cores?

Section 5.2.3 showed that depending on the active core location, temperature and power consumption can be severely affected, even though the *same* number of cores is used for execution. We project that this will become more apparent in the architecture with many more cores. For example, NVidia GTX280 has 30 SMs, compared to 12 SMs of 8800GT. This GPU will give us more room to choose the number of active cores and their locations. However, we do not have the thermo-spacer and leave this for future work. Nevertheless, we have successfully done a similar experiment using the on-chip temperature sensor and power meter. The plot of measured power and temperature is shown in Figure 11. It shows a significant delta of around 25 watts with a temperature differential of 4.3 degree Celsius. This work adds one more dimension, controlling core locations, to this work [6], which claims that not all cores need to be activated for some benchmarks.

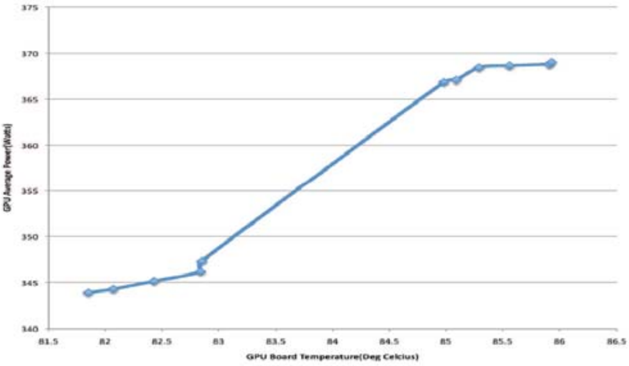


Fig. 11. Variation of GPU power consumption vs. on-chip temperature.

6. Implications and Future Work

The results in Section 5.2.3 showed that the number of TPCs activated should be minimized to reduce power and temperature. To avoid confusion, minimizing the number of active TPCs is not the same as minimizing the number of active cores; this is transparent to a programmer. Another implication is that those active TPCs should be as far apart as possible. This fact was actually considered in [8]; if hot and cold areas are interchangeably placed, they create a virtual heatsink effect. The difference is that the authors used a simulator, and the granularity of control was different (i.e., controlling CPU units vs. GPU cores). We actually controlled scheduling at core and TPC granularity and confirmed this effect in a real experiment.

To maximize the virtual heatsink effect, separating the active TPCs as far apart as possible is clearly the method to use. However, there is a complicated trade-off between (1) maximizing all SMs in a single TPC vs. (2) minimizing the number of active SMs in a TPC, which results in more active TPCs. It would seem that activating all SMs within the same TPC is better since SMs share some texture and shared cache. Furthermore, activating another TPC unnecessarily could result in more energy use. But there could be another trade-off. For some types of applications with heavy memory use, SMs in the same TPC could compete with each other for memory load/store units, which could degrade performance. For this case, invoking multiple TPCs would result in better performance. This deep level of investigation is future work. Nevertheless, we have managed to measure the temperature of a GPU processor and perform explicit work scheduling despite having no disclosed information from vendors.

To the best of our knowledge, this is the first study to analyze the thermal behavior of a GPU processor using thermocouples and it extends [6] by adding one more dimension of energy optimization, which is changing active core location, not just limiting the number of cores.

7. Conclusion

In this paper we present a robust and reliable temperature-measurement system using thermocouples. Furthermore, we overcome the GPU scheduling problem despite the lack of documentation on scheduling.

We discuss the importance and an application of such a system by describing its relevance to a thermal-aware scheduling scheme for many-core systems. With power and temperature having become primary-level design parameters and with the advent of many cores, we believe that this field of research offers many opportunities for exploration and needs robust tools to achieve that exploration. To this effect, we feel that the system described in this paper will prove to be very beneficial.

References

[1] Extech 380801. http://www.extech.com/instrument/products.

[2] National instruments fp-tc 120 thermocouple data logger. http://sine.ni.com/nips/cds/view/p/lang/en/nid/2187.

[3] NVIDIA GeForce series GTX280, 8800GTX, 8800GT. http://www.nvidia.com/geforce.

[4] Nvidia's geforce gtx280 graphics processor. http://techreport.com/articles.x/14934/2.

[5] M. Floyd, S. Ghiasi, T. Keller, K. Rajamani, F. Rawson, J. Rubio, and M. Ware. System power management support in the ibm power6 microprocessor. *IBM Journal of Research and Development*, 2007.

[6] S. Hong and H. Kim. Ipp:an integrated gpu power and performance model that predicts optimal number of active cores to save energy at static time. In *ISCA '10: Proc. of the 37th annual Int'l. Symp. on Computer Architecture*, 2010.

[7] 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 *ISPASS '09: Proc. of the IEEE Int'l. Symp. on Performance Analysis of Systems and Software*, 2009, April 2009.

[8] W. Huang, M. R. Stant, K. Sankaranarayanan, R. J. Ribando, and K. Skadron. Many-core design from a thermal perspective. In *DAC ' 08: Proc. of the 45th conference on Design automation*, 2008.

[9] LAVA research group. Hotspot. http://lava.cs.virginia.edu/HotSpot/index.htm.

[10] F. J. Mesa-Martinez, E. K. Ardestani, and J. Renau. Characterizing processor thermal behavior. In Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems. In *ASPLOS*, 2010.

[11] F. J. Mesa-Martinez, J. Nayfach-Battilana, and J. Renau. Power model validation through thermal measurements. In *ISCA '07: Proc. of the 34th annual Int'l. Symp. on Computer Architecture*, 2007.

Aniruddha Dasgupta earned his Bachelor of Engineering in Electronics and Telecommunications from Pune University, India, in 2006. Thereafter he earned a master's degree in electrical and computer engineering from Georgia Institute of Technology, USA, in 2011. During the course of his master's he did research work on CUDA performance analysis. Currently he works at AMD, Austin for the Power and Performance Optimization Labs. His responsibilities at AMD include post silicon tuning of power and performance of AMD CPUs and APUs.

Sunpyo Hong received BA and MS degrees in electrical and computer engineering from the Georgia Institute of Technology, where he is currently working toward a doctoral degree in computer engineering. His research interests include energy-efficient many-core architectures and GPGPU computing, with a focus on analytical and empirical modeling of performance and power using architectural characteristics and compiler-architecture interaction. He is a student member of the IEEE and the ACM.

Hyesoon Kim: Dr. Kim is an assistant professor in the School of Computer Science at Georgia Institute of Technology. Her research interests include high-performance energy-efficient heterogeneous architectures, programmer-compiler-microarchitecture interaction. She received a B.S. in mechanical engineering from Korea Advanced Institute of Science and Technology (KAIST), an M.S. in mechanical engineering from Seoul National University, and an M.S. and a Ph.D. in computer engineering at The University of Texas at Austin.

Jinil Park: Dr. Park is a professor in the School of Mechanical Engineering at Ajou University, Suwon, Korea. His research interests include automotive engineering and thermo-fluidic measurement. He received a B.S. and a Ph.D. from Seoul National University. He was a research fellow at Brown University from 2000 to 2003 and a research fellow at Tokyo University from 2003 to 2004 before joining Ajou University.