Temperature-Aware GPU Design

Jeremy W. Sheaffer, Kevin Skadron, and David P. Luebke
University of Virginia Dept. of Computer Science
{jws9c, skadron, luebke}@cs.virginia.edu

The Need for Temperature-Aware Design

Cooling for graphics processors is becoming prohibitively expensive. Even for GPUs not intended for high-performance markets, cooling is a serious issue due to the low profit margins in these market segments. Much of the heat originates from the processor core itself. This paper argues for a *runtime* approach to cooling, reducing the need for bulky and expensive thermal packages and fans. Today's cooling solutions are designed for worst-case behavior. First, localized heating occurs much faster than chip-wide heating; since power dissipation is spatially non-uniform across the chip, this leads to "hot spots" and spatial gradients that can cause timing errors or even physical damage. Reducing these hot spots, whether through changes in circuit design, microarchitecture, or software, will help reduce cooling requirements. Second, the package should be designed for the worst *typical* application. True worst-case behavior is rare, and a solution designed for worst case is in fact overdesigned for most typical operating conditions. However, a package designed for typical behavior could be overcome by some unusual application, and so should engage *dynamic thermal management* (DTM). These techniques throttle back the chip's power dissipation (and possibly performance) until the thermal stress has passed. DTM has recently been the subject of considerable research in the general-purpose computer-architecture community, and it is used in commercial chips like the Pentium 4, Pentium M, and Transmeta Crusoe.

It is important to note that runtime thermal management cannot merely be achieved by designing the chip for greater energy efficiency. Thermal behavior evolves over time scales of hundreds of microseconds or milliseconds. This means that power-management techniques, in order to be used for thermal management, must directly target the spatial and temporal behavior of operating temperature. In fact, many low-power techniques have insufficient effect on operating temperature, because they do not reduce power density in hot spots, or because they only reclaim slack and do not reduce power and temperature when no slack is present.

DTM for GPUs

To study thermal issues in a GPU, we have developed a simulator called *Qsilver* that models GPU clock-cycle-by-cycle activity and power in the microarchitecture domain. Qsilver uses the *Chromium* system (`http://chromium.sourceforge.net/`) to intercept a stream of OpenGL calls which it traces through the simulator. We augment Qsilver with an architectural thermal model called HotSpot (`http://lava.cs.virginia.edu/HotSpot/`) that tracks temperature in each unit over time.

Using the game *Enemy Territories: Return to Castle Wolfenstein* as a sample application and a hypothetical low-end GPU modeled after nVIDIA's GeForce4 (but adapted for low-resolution console use), we characterized the thermal behavior in each architectural unit over 25000 cycle sampling intervals. We modeled a fan-less aluminum cooling solution with a maximum specified operating temperature of 105°C, and tested three different DTM techniques: *dynamic voltage scaling* (DVS), *clock gating*, and *toggling* (also known as *fetch gating*). DVS reduces voltage and frequency and also leakage power, but entails some stall time while clock circuitry resynchronizes. Clock gating simply freezes the clock momentarily. Toggling reduces the duty cycle at which vertices are transformed. Neither clock gating nor toggling reduces leakage.

Our simulator is tuned to model a typical console architecture driving an 800×600 pixel display, with a processor core on an 0.18 micron process running at 1.8V and 300MHz, an aluminum cooling solution, and no fan. Over the course of a typical 50 frame (1.6s) sequence from *Enemy Territories*, the chip exceeds 105°C, and runs at over 100°C for 90% of the cycles. Employing DTM techniques, we are able to significantly reduce these numbers.

While utilizing clock gating at a 100°C threshold, the chip never exceeds 100.03°C, and only runs at above 100°C 16% of the time. This solution incurs a 19% performance penalty. Employing toggling on the vertex engine, allowing the unit to operate only 1 out of 4 cycles while DTM is engaged, yields a high temperature of 102.97°C with only 14% of cycles exceeding the 100°C threshold and a 17% performance hit. DVS tends to be less heavy-handed in our experiments. Scaling the voltage down by 20% correspondingly scales the frequency to 253MHz when the chip is above the threshold temperature. The processor reaches a maximum temperature of only 100.17°C, with 24% of cycles exceeding the 100°C threshold, but performance is penalized by only 4.6%.

On the floorplan used for the above experiments (modeled after a marketing figure of the GeForce4 floorplan), the two hottest functional units, the framebuffer operations unit and the vertex engine, are located next to each other across the top of the chip. We designed another floorplan, only slightly modified from this design, which permutes the right side of the chip by moving a cool unit, 2D video operations, up next to framebuffer ops, and moving the vertex engine down, thus placing the two hotspots on opposite corners of the silicon. This change had the effect of bringing down the maximum temperature, sans DTM, of framebuffer ops from 105.61°C to 105.11°C, and of the vertex engine from 102.45°C to 101.23°C, and reducing performance penalties to 15.64%, 13.41%, and 3.73% for clock gating, toggling, and DVS respectively, improvements of 3.58%, 3.34%, and 0.87%.

These early results show the potential of runtime, graphics-architecture techniques for managing operating temperature and reducing cooling costs. Dynamic voltage scaling is particularly attractive given the small performance penalty evident in our experiments, and can be even more attractive if the architecture implements independent voltage and clock domains. This allows voltage and frequency in different portions of the architecture to be controlled independently according to activity levels. This allows thermal control to be combined with substantial energy savings of approximately 20%, and even more in future process technologies.