Keck Next Generation Adaptive Optics

Real-Time Controller Mini-Design Review

December, 2009

Response to Reviewer Comments

Tuan Truong – All Documents

1. It is not clear from the RTC Design document (section 1.8.4) that "the Tomography Engine can only meet requirements if it is implanted with a multiple card system based on FPGAs." It would help to see a convincing argument against GPU-based solutions.

> We did an analysis of the GPU option and it appears that it has trouble with I/O rates of the amount envisioned.

2. Appendix B shows unusually high bus transfer times between host CPU memory and GPU device memory that are inconsistent with our experience on PALM-3000 based on the 8800 Ultra and GTX GPU series. The matrix and pixel transfer times seem to be about 8 to 10 times as slow. I didn't check other times listed. Also, your device-to-host and host-to-device peak rates in the Transfer Rate vs. Data Size chart seems twice as slow.
My guess is we are probably using different motherboards.

> The table and figure in Appendix B are for "unpinned" memory. We've since determined that "pinned" memory gets about a 2x speedup from this. This is PCIe-8, PCIe-16 would give further benefit, but in the end it doesn't meet the thoughput requirements for the tomography engine.

3. Figure 24 seems to suggest centroiding is done by the host CPU. If
so, you may see occasional jitter up to 30us, if not more, in the compute latency. The frequency and magnitude of jitter can be a major source of headache, even when running RT Linux, at least with SUSE RT in our experience. I'd suggest this variation be quantified early in your RT Linux selection. We see it with any sub-ms calculation, not just centroiding, when executed on the host CPU. In your case, you may see it also with force/disp calculation and the command generation step that follows. Thus you should weight the option of doing both centroiding and reconstruction on the same FPGAs or GPUs.

> No, all the processor blocks in Figure 24 are done on the FPGA.

4. Figure 47 neglects to show in the Offload Processor diagram the necessary frame grabbers to capture the camera data on behalf of the disk subsystem. If telemetry data are first written to host memory and then read back for transfer to the disk subsystem, the total data rate will be 2 x 1.868 MB/sec, not including the RTC ethernet traffic nor memory access due to the data sampling/filtering done by the host CPU. At such I/O rate, it will be a challenge to build the Offload Processor out of a single Linux box, be it multi-processor multi-core or not. By the way, you should consider moving figure 46 toward the front of the document. as it provides the most descriptive view of the RTC system in my opinion.

> The camera data is split and goes in parallel to both the wavefront sensor processors and to the offload processor. There is no host computer involved; this is all done in FPGAs, up to the Fiberlink interface to the RTC Disk Subsystem.

5. Regarding the sub-ms accurate GPS-based clock used to time stamp telemetry and diagnostics data, it would be helpful to show its physical connection to other required subsystems. In your selection of such clock, it is important to consider the ease of programming and the overhead associated with retrieving the current time.

> The GPS time source is queried a few times per second to synchronize the internal time-stamp, which is distributed, in parallel, through high-speed serial links to the FPGAs that are streaming the telemetry data. Within this dedicated telemetry FPGA, a Fits header is periodically attached that contains the time stamp. The header also contains all the current configuration data, which on the frame time scale is static and thus pre-stored in this same FPGA.

One minor point: Tables 2 and 6 show 450 us while figures 12, 13 and 17 assume 400 us.So it is unclear what the tomography latency is exactly.