Review of NVIDIA Tesla: A Unified Graphics and Computing Architecture [1]

Riley Wood

April 5, 2016

Key Ideas

The authors open by discussing the evolution of two different types of GPU cores: vertex and pixel processors. Each one was initially optimized differently, but lately their designs have been converging. The authors cite how this convergence has led to an increase in design complexity that could be mitigated by merging the two processor types, which is just what the Tesla architecture does. Additionally, different workloads will demand each processor resource in varying proportions, meaning it is difficult to ensure that the load on processors is always balanced – we would prefer to dynamically balance load. They cite these motivators as reasons why they unify the two processor types as one in the Tesla. From the block diagram, one can see blocks dedicated to managing the distribution of pixel and vertex work to the same set of processors. The processors available are partitioned into clusters, and then each cluster is organized into two streaming multiprocessors, each made up of eight streaming processors. They discuss the multithreading capability of the multiprocessors and introduce the concept of SIMT (single-instruction multiple-thread) scheduling and warps, which are groupings of 32 instructions that can execute at once.

Review

I would like to see a comparison of vertex and pixel workloads on the Tesla's unified processors versus on another architecture with separate vertex and pixel processors. I would be interested to see if the Tesla falls short in some tests because of its lack of specialization, i.e. maybe its cores can't do vertex/pixel computations as well as processors built to do only those computations. Something I sought clarification on was how having warps differs from having a 32-issue instruction fetch where each reservation station is for a separate instruction stream. I suppose a 32-issue machine is still sharing the same datapath among its instructions, whereas here the instructions execute truly in parallel, not just intermixed. Something else that could use clarification is the memory hierarchy. All SPs within a multiprocessor have shared memory. I assume each SP has its own L1 cache to avoid dipping into larger, slower, shared memory. So is the memory shared amongst SPs somewhere between an L1 and an L2? Because the L2 appears in segments connected to the TPCs via a network.

Conclusion

References

[1] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. Nvidia tesla: A unified graphics and computing architecture. *IEEE Micro*, 2008.