**G³ (gem5 + GPGPU-Sim): A Simulator for Heterogeneous Processors**

Marc Orr, Jason Power & David Wood

**Motivation**
- Asymmetric chips accelerate sequential and parallel code
- Hybrid CPU-GPU designs (e.g. AMD Fusion, Intel Sandy Bridge) are being manufactured today
- How can we simulate these architectures?

**CUDA Example**
```c
char *host_data = malloc();
char *device_data = cudaMalloc();
//copy host data to device
cudaMemcpy(host_data, device_data);
//launch kernel
kernel(device_data);
//copy device data back to host
cudaMemcpy(device_data, host_data);
```

**G³ Features**
- Cache coherence protocols
  - “In-coherent” – Like NVIDIA Fermi
  - Heterogeneous – Future architectures
- Times host-device memory copies
- Models driver delays
- Asynchronous kernel launches
- Models atomic and uncached memory requests

**Current Work**
- Cache coherence in a unified address space
  - Hides average of 90% of memory copy
  - GPU can tolerate cache-to-cache latency
  - Simplifies programming
- Heterogeneous (CPU/GPU) workloads
  - Very hard to program and generalize
  - None are widely available today
- Synchronization on GPGPUs
  - Due to high cache latencies, sync is a bottleneck in DLP workloads.
  - Possible solutions: Hardware barrier or redesigning GPU cache hierarchy
  - What does memory consistency look like in a unified address space?