Abstract

- GPGPU-Sim, Multi2Sim, Barra (GPU simulators)
- It can run unmodified OpenCL-based applications

Introduction

- CPU tools
    - Transient faults on hardware structures
    - Noise
    - Hardware reliability
    - OpenRISC
- RTL-level implementations and low-level microarchitecture are lacking
- Goals
    - Realism
    - Flexible
    - Can run OpenCL or CUDA
    - M0, 1, 2 - memory controllers
- Non-goals
    - Not ASIC compatible
- Commonalities between AMD and Nvidia
- Ideas
    - Physical design perspective to “traditional” microarchitecture research
    - Thread-block compaction - warp scheduler technique
    - Circuit-failure prediction
    - Timing speculation
    - Transient fault injection

MIAOW Architecture

- ISA
    - Program counter
    - Execute mask
    - Status registers
    - Mode register
    - General purpose registers

- MIAOW Processor Design Overview
  - Has host CPU
  - Assigns kernel to GPGPU
    - Handled by GPU's ultra threaded dispatcher
    - Computes kernel assignments and schedules wavefronts to CUs
      - Allocates wavefront slots, registers, and LDS space
    - CU execute the kernels
      - scalar and vector ALUs
      - load-store unit
      - **LDS (Local data store) is internal scratch pad memory**
      - have access to device memory through memory controller
    - L1
      - scalar data accesses and instructions
    - Unified L2
  - You can schedule up to **40 wavefronts** on each CU
- Compute Unit Microarchitecture
  - Fetch - interface between dispatcher and compute unit
    - Receives initial PC value
    - Range of registers and local mem it can use
    - Unique ID for wavefront
    - Also informs dispatcher when wavefront is completed
  - Wavepool - Instruction queue for fetched instructions
    - Up to 40 wavefronts - 40 independent queues
    - Can reside in compute unit at a time
  - Decode - instruction
    - Collates 2 32bit-halfs of 64bit instructions?
  - *Workgroup - more than 1 wavefront, but mapped to a single CU*
  - Issue/schedule - tracks all in-flight instructions
    - Scoreboard to resolve dependencies
    - Handles barrier and halt instructions
  - Vector ALU - 16 wide (processed in 4 batches)
  - Register files
    - in CU - 1024 vector registers

        - 512 scalar registers
        - Vector register files in 64 pages / banks
            - Page corresponds to each thread in wavefront
- Ultra-threaded dispatcher
    - Global wavefront scheduling
    - Receives workgroups from host CPU, passes them to CUs
    - LDS, GDS, and register files never overlap
    - Workgroups come through host interface, if workgroup table available, accept
        - Otherwise, let the host know it can’t
    - -> Goes to **resource allocator**
        - finds a CU that has enough resources
        - If free CU found, CU id and allocation data are passed to resource table
    - GPU interface
        - workgroup into wavefronts to the CU
- Design choices
    - Fetch takes in 1 at a time. GCN does 16-32, per cache line