SIMT Control & Data Flow Representation

Discussion Group Outbrief

Scalable Tools Workshop

August 15th, 2024

Attendees

Barton Miller, Hsuan-Heng Wu, Ben Woodard, Ronak Chauhan, Benjamin Welton, Timour Paltashev

Remote AMD attendees : Lancelot Six, Scott Linder

Notes : Sébastien Darche

What *is* SIMT ?

- SIM*D* instructions use predicated operation
- SIM*T* model relies on a execution mask. Instructions are executed (written back) on all active lanes
- SIMT model has a significant influence on how we define control flow

Scalar / vectorized CFG

- If all lanes are convergent, scalar & "vectorized" control and data flow are similar
- Per-lane control flow analysis depends on the exec mask and where it was stored
- Scalar unit basic block may be split if the exec mask changes
  - Recognize instructions manipulating the execution mask
  - Is a scalar operation modifying the mask its own basic block (e.g. flipping for if/else)
- What about compiler-inserted whole-lane operations (spilling?)
- Support for reachability analysis

Data flow

- Common case : everyone stays in their lane. Represented by a scalar, but is actually a vector
- What happens when you have cross-lane operations (scalar ops, LDS)
    - Data flow graph explodes
- Can it be represented concisely?

DWARF info on heterogeneous source

- Location description
- Can provide some insights on divergent control flow from a "context"
  - Scalar PC
  - Execution mask
- `pushlane` operator for data flow ?
- Used by `rocgdb`, e.g. to compute current lane PC
- Some will be implemented in the upcoming DWARF 6 standard
- See : LLVM doc on DWARF extensions for heterogeneous debugging[1]

[1] `https://llvm.org/docs/AMDGPUDwarfExtensionsForHeterogeneousDebugging.html`