Introduction of Architecturally Visible Storage in Instruction Set Extensions

Author: Partha Biswas, Nikil D. Dutt, Laura Pozzi, Paolo Ienne
Presentation by Ganghee Jang
Overview

- What is ISE & Architecturally Visible Storage?
- Motivation
- Related Work
- Design Approaches with AVS
- Results
- Not Bird’s eyes – Different View with my Eyes
What is ISE & AVS?

- **ISE (Instruction Set Extension)**
  - Instruction Set to support Customized Functional Unit

- **AVS (Architecturally Visible Storage)**
  - Visible Storage from Functional Unit (Architecturally)

- From my Understanding,
  - "Visible" means "Where" data is Synchronized to
  - To have Ownership of Processing Data, FU can have its own state and internal data
  - Access Memory or Cache Directly
  - Internal Memory
Motivation

- Let’s assume an Instruction multiplying a Scalar and Matrix
- Mult_s_M(scalar, Matrix)
Motivation (Cont’)

- The Benefit of AVS
- Lowers Cache Pollution
  - Data bypassing Cache and Register
- Increase Scope of ISE Algorithm
  - Including load/store Instructions give Extra Chances for more Optimization of Algorithm
- Reduce Energy Consumption
  - By reducing Memory Access
Related Work

- Scratchpad
- Avoid some Cache Pollution
- Still share Register File
Related Work (Con’t)

- Intelligent SRAM
- Move ALU nearer to On-Chip SRAM
- Increase Complexity of Identification Problem
Design Approaches without AVS

- Cut: Identified codes for ISE
- All Arguments for AFU from Register File
Design Approaches with AVS

- Increased Cut, so Increased Granularity
- Need DMA for efficient Data Transfer
CFG & Basic Block

- CFG: Control Flow Graph
  - To represent Program as a graph

- Basic Block: Each Node in the CFG
  - Piece of code which has following Characteristics
  - Entrance: only at the Beginning
  - Exit: only at the End
A node $i$ dominates node $j$ if every path from the entry node to $j$ passes through $i$. 
Where to put DMA Instructions?

- CFG of fft Algorithm
- P: last Polluter to cbb
- cbb: refers to the basic block for which an ISE is currently being identified
Where to put DMA Instructions?

- R: Nodes can be reached from p
- D: Nodes strictly Dominates cbb
- Candidates: \{4, 5, 6, 7, 8, 9\}
- Select the Node with the Least Execution Count

Candidates: \( R \cap D \)
- no MEM: without allowing memory inclusion
- w/ VEC: local memory inside with vector access
- w/ VEC + SCA: local memory inside with vector access and scalar access
Results (Cont’)

- One of the good Example of H/W, S/W co-Design
- Solving a Problem in ISE Design Domain
- Speedup
  - Memory Wall Problem by Internal Memory
  - Bigger Granularity
My Observation

- Hidden Assumption
  - No or very small Dependency between successive AFU Operation
    - DMA can hurt Performance
  - Granularity
    - All AFU Operation need bigger Granularity?
  - Compiler must know all the detail about DMA latency
    - Need Profiling and recompile
    - Is it feasible with Reconfigurable ISE?
Question?