Polymorphic Architecture

Honggo Wijaya and Jing Jiang

Introduction : Motivation

- Two conflicting requirements :
  - Narrowly focused custom solutions for high performance and high efficiency
  - High volumes of widely applicable general purpose designs to accommodate design time and high non-recurring costs
- Wire delay cause scalability issues
- Solutions : Modular Polymorphic Architecture

Smart Memories : Overview

- Modular Architecture with coarse grained reconfigurability
- Tailor the appearances of on-chip memory, interconnection network, the processing elements
- Array of processor tiles and on-die DRAM memories with dynamically routed network.
- Cluster four tiles into “quad”
  - More computation power
  - low-overhead, intra-quad interconnection
  - reduce the global interconnection

Smart Memories : Tile Architecture

- Reconfigurable memory system
  - different applications have different memory access patterns
  - streaming mode, coherence protocol
- Crossbar Interconnection
  - Connect memory mats to processor or quad interface port
- Processor
  - Integer and FP clusters
  - local register files and shared FP register file
  - Reconfigurable instruction format/decode
  - Sustain two independent threads in each tile

Smart Memories : Results

- Mapped really well to two different architectural spectrum :
  - Imagine : streaming processor
  - Hydra : speculative multiprocessors

Smart Memories : Strengths and Weaknesses

- Strengths :
  - modular design, able to map different architecture with little degradation
- Weaknesses :
  - Mapping complexity

TRIPS: Overview

- 4 out-of-order, 16-wide-issue Grid processor cores, which can be partitioned
- software schedulers optimized for point to point communication
- block oriented in all modes of operation, programs compiled into large blocks of instructions with a single entry point, no internal loops, and possible multiple exit points as found in hyperblocks.

TRIPS: Hyperblocks

- compiler responsible for statistically scheduling each block of instruction onto computation engine
- each block has a set of state inputs and a potentially variable set of state outputs that depend upon the exit point from the block
- run time sequence –fetch a block, load to the engine, execute it to completion, committing its results to the persistent architecture state and proceed to next block

TRIPS: Polymorphous resource

- Frame space – reservation stations with the same index across all nodes
- Register file banks –extra copies used for speculation or multithreading, depending on mode of operation
- Block sequencing controls – various policies for different modes, Eg. Deallocation logic maybe configured to allow a block to execute more than once, as is useful in streaming applications
- Memory tiles – scratch pad memory, synchronization buffers etc

TRIPS:Strength and Weakness

- Strengths:
  - Have different modes for DLP, ILP, TLP
  - Can have a mix of those
- Cons:
  - No modeling of page fault
  - No issue of wrong path instruction to the memory
  - Complex architecture due to speculation
  - power issue

Conclusion

- Performance and efficiency of custom solutions are still better.
  - There's always trade off between flexibility and performance.
- Modular architecture solves scalability issues
- Major challenges:
  - Design interfaces between software and hardware
  - Determine how and when to initiate reconfiguration

Discussion

- How fine grain polymorphic architecture should be? Build a huge core and break it up if needed or build many small cores and connect them if you need it ?
- Differences between TRIPS and Smart Memories ? Approach to power and complexity? The way they handle DLP/TLP/ILP?
- Any Similarities ?
- Smart Memories:
  - How do you support cache coherence in Smart Memories for configurations larger than those in the paper? How do you support thread speculation ?
- TRIPS:
  - What are the crucial mechanisms?
- How much support for ILP/DLP/TLP is needed in a polymorphic processor? How much of it in HW and how much of it in SW? Pros and Cons of HW and SW? Implications for programming model and compiler for each approach.

Discussion (cont.)

- Can we decide the polymorphous configurations automatically based on the applications? Dynamically or statically?
- Is it possible to dynamically switch mode if the application has different parallelism at different phase? What processor has this capability?
- Can we change the clock frequency dynamically based on the current workload?