The CAE Architecture: Decoupled Program Control for Energy-Efficient Performance

Ronny Krashinsky and Michael Sung

- Change in project direction from original proposal
  - initial idea of energy-efficient program control used a lot of existing ideas already
- New idea of combining existing work in decoupled architectures to simplify program control and issue logic for high-performance microprocessors
- Basic idea is that program control is inherently separate from other types of instructions (access and execute). We propose to decouple the program control from the rest of the instruction stream(s) to uncover ILP

Prior Work

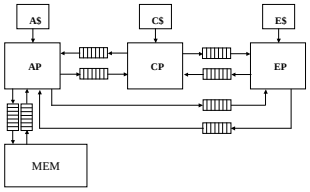
- Separation of access and execute functions
  - IBM 360/370, CDC 6600, CDC 7600, CRAY-1
- Explicit partition of access and computation functions
  - J. Smith, PIPE (compile time splitting)
  - G. Tyson, MISC (Multiple Inst. Stream Computer), descendant of PIPE
  - A. Pleszkun, SMA (Structured Memory Access) architecture
  - J. Smith, Astronautics Corporation's ZS-1 (splits fix-point/addressing from floating-point operations)
  - A. Wulf, WM architecture
  - IBM's FOM (FORTRAN Oriented Machine)
- Shares similarities with trace processors
- Technically, superscalar machines have a slightly decoupled nature

Motivation for Decoupled Architectures

- Increased performance by exploiting fine-grained parallelism between access and execute functions
  - Decoupling access from execution allows access processor to run ahead or "slip" w.r.t. execution processor: dynamic reordering
  - Memory latency toleration
  - Dynamic loop unrolling (exposing ILP between loop iterations, hide functional unit latencies by overlapping executions of diff. iterations)
- Simplified issue/decode logic
  - Much simpler than complex superscalar architectures (IW, ROB, bypass)
- Scalability – direct consequence of simplified logic
  - For superscalar processors, need to increase IW which does not scale (Palacharla/Agawal papers)
  - For decoupled machines, simply lengthen queues to allow more "slip"

Decoupled Program Control: CAE Architecture

- Hierarchy of decoupling: 3 levels of decoupling (Control, Access, and Execute).
  - Control flow is most elastic, providing ample instructions for both access and execute pipelines



Decoupled Program Control Benefits

- Program control flow can be easily determined without waiting
- Provides "out-of-order" execution without complexity
- Inherits the memory latency toleration of DAE architectures
- Simplified issue logic
  - Can be implemented with small structures/queues
- Allows non-speculative instruction prefetching
  - Because of prefetching, we can shrink data structures like caches, potentially reducing critical paths as well as reducing power
  - Still provides performance advantage for streaming applications, etc. where lack of locality mitigates performance advantages of caches

Progress and Roadmap

Completed

- Literature search of decoupled architectures
- Initial ISA exploration and microarchitectural development for proposed CAE architecture

Roadmap

- Classifying control flow and dependencies in applications
- ISA development for each instruction stream (control, access, execute)
- Complete architectural specification of CAE architecture (TRS)
- Implementation of RTL-level simulator (SyCHOSys)
- Simulation and performance analysis (SyCHOSys)

Issues with Decoupled Program Control

- **Deadlocking with queues**
  - Queue size determines how much slip can occur
  - Draining queues or explicit queue manipulation (push/pop) instructions
- **Performance issues from feedback of values from execution/ access pipelines that program control depends on**
  - Dependencies limit how far the program pipeline can slip in front of the access and execute pipelines, etc.
  - Likewise, feedback dependencies from execute to access pipelines
- **Complexities in queue interactions (correctness, verification, ease of programming)**
  - Basically an issue of how to to synchronize instruction streams correctly

The CAE Architecture: Decoupled Program Control for Energy-Efficient Performance

Ronny Krashinsky and Michael Sung

- Change in project direction from original proposal
  - initial idea of energy-efficient program control used a lot of existing ideas already
- New idea of combining existing work in decoupled architectures to simplify program control and issue logic for high-performance microprocessors
- Basic idea is that program control is inherently separate from other types of instructions (access and execute). We propose to decouple the program control from the rest of the instruction stream(s) to uncover ILP

Prior Work

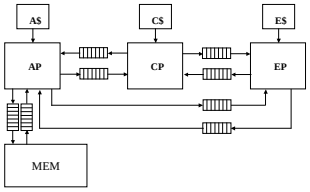
- Separation of access and execute functions
  - IBM 360/370, CDC 6600, CDC 7600, CRAY-1
- Explicit partition of access and computation functions
  - J. Smith, PIPE (compile time splitting)
  - G. Tyson, MISC (Multiple Inst. Stream Computer), descendant of PIPE
  - A. Pleszkun, SMA (Structured Memory Access) architecture
  - J. Smith, Astronautics Corporation's ZS-1 (splits fix-point/addressing from floating-point operations)
  - A. Wulf, WM architecture
  - IBM's FOM (FORTRAN Oriented Machine)
- Shares similarities with trace processors
- Technically, superscalar machines have a slightly decoupled nature

Motivation for Decoupled Architectures

- Increased performance by exploiting fine-grained parallelism between access and execute functions
  - Decoupling access from execution allows access processor to run ahead or "slip" w.r.t. execution processor: dynamic reordering
  - Memory latency toleration
  - Dynamic loop unrolling (exposing ILP between loop iterations, hide functional unit latencies by overlapping executions of diff. iterations)
- Simplified issue/decode logic
  - Much simpler than complex superscalar architectures (IW, ROB, bypass)
- Scalability – direct consequence of simplified logic
  - For superscalar processors, need to increase IW which does not scale (Palacharla/Agawal papers)
  - For decoupled machines, simply lengthen queues to allow more "slip"

Decoupled Program Control: CAE Architecture

- Hierarchy of decoupling: 3 levels of decoupling (Control, Access, and Execute).
  - Control flow is most elastic, providing ample instructions for both access and execute pipelines



Decoupled Program Control Benefits

- Program control flow can be easily determined without waiting
- Provides "out-of-order" execution without complexity
- Inherits the memory latency toleration of DAE architectures
- Simplified issue logic
  - Can be implemented with small structures/queues
- Allows non-speculative instruction prefetching
  - Because of prefetching, we can shrink data structures like caches, potentially reducing critical paths as well as reducing power
  - Still provides performance advantage for streaming applications, etc. where lack of locality mitigates performance advantages of caches

Progress and Roadmap

Completed

- Literature search of decoupled architectures
- Initial ISA exploration and microarchitectural development for proposed CAE architecture

Roadmap

- Classifying control flow and dependencies in applications
- ISA development for each instruction stream (control, access, execute)
- Complete architectural specification of CAE architecture (TRS)
- Implementation of RTL-level simulator (SyCHOSys)
- Simulation and performance analysis (SyCHOSys)

Issues with Decoupled Program Control

- **Deadlocking with queues**
  - Queue size determines how much slip can occur
  - Draining queues or explicit queue manipulation (push/pop) instructions
- **Performance issues from feedback of values from execution/ access pipelines that program control depends on**
  - Dependencies limit how far the program pipeline can slip in front of the access and execute pipelines, etc.
  - Likewise, feedback dependencies from execute to access pipelines
- **Complexities in queue interactions (correctness, verification, ease of programming)**
  - Basically an issue of how to to synchronize instruction streams correctly