Outline

- Microarchitecture
- Memory Hierarchy
- Multi-processor
- References
- Q&A
Microarchitecture

- **Thread modes**: 1 / 2 / 4 / 8
- **Pipeline units**
  - Fetch Unit
    - Fetch 8 / cycle
    - Decode 8 / cycle
  - Sequencing Unit
    - Dispatch 8 / cycle
    - Issue 10 / cycle
  - Load / Store Unit, Execution units
    - Execute 10 / cycle
    - Commit 8 / cycle
Microarchitecture: Fetch Unit stages

- Unified L2
- Predecode
- I-Cache
- Instruction buffer
- Group formation
- Decode
- Fusion
Microarchitecture: Predecode

- Mark branches w/ branch prediction
  - Unconditional / Predict taken branches
  - Predict direction and target address and send to PC for next fetch cycle
  - Takes three cycles
  - In SMT overlap the two wasted fetch cycles
Microarchitecture: Instruction buffer

- Fetch instructions from I-Cache
- One of PC’s is picked for fetching from I-Cache
  - Programmable thread priority registers
  - Partition fairness
  - Instruction buffer fullness
Microarchitecture: Group formation

- From Instruction buffer, select
  - 1 group in ST
    - 6 non-branches and 2 branches
  - 2 groups in SMT
    - 3 non-branches and 1 branch

- Groups are now the atomic unit for Sequencing Unit
Microarchitecture: Decode and Fusion

- Decode
  - Simple instructions are dispatched
  - Complex instructions are cracked and dispatched

- Fusion
  - Conditional branch followed by a simple instruction
    - Simple: Fixed-point / Store
    - Remove conditional branch
    - Make simple instruction predicated on that condition
Microarchitecture: Sequencing Unit

- Reservation stations
- Register files
- Issue queues
Microarchitecture: Reservation stations

- Retire a group
  - All instructions in a group finish
  - And the group is the oldest in its thread

- Flush a group

- Flush a mispredicted branch
  - When it’s the first branch
    - It’s not the last instruction in the group
    - Partially flush subsequent instructions
  - When it’s the second branch
    - It will always be the last instruction
    - No need to flush the group
Microarchitecture: Register files

- Register renaming instead of reorder buffer
- Architected registers (R0 ~ R31) and non-architected registers
- Two levels
  - Except for SMT8
    - All non-architected registers and all architected registers
  - SMT8
    - All non-architected registers and some architected registers
    - The rest of architected registers
Microarchitecture: Issue queues

- UniQ
  - Most types of instructions
  - Large
  - Non-shifting
  - Each entry with a position register
  - Reduce power while not sacrificing capacity
Microarchitecture: Issuing

- Priority for oldest
- Within a priority set, all can issue out-of-order
- In UniQ, fixed-point can also be issued to Load / Store Unit
Microarchitecture: Load / Store Units

- Load / Store Reorder Queues
Microarchitecture: Load / Store Reorder Queue

- Virtual space > Physical space
- Tag = Pointer into the queue’s virtual space
- Interactions with UniQ
  - Virtual tag assigned and comes into UniQ
  - Wait for free physical space
  - Convert to real tag and come into physical space
  - Wait for dependency clear from UniQ
  - Issue to downstream
  - Upon completion, free physical space for next virtual tag
Memory Hierarchy

Memory Hierarchy

- **On core L1 cache**
  - 32 KB, 8-way set associativity Instruction cache
  - 64 KB, 8-way set associativity Data cache

- **On chip Unified L2 cache**
  - 512 KB SRAM, 8-way associativity

- **On chip Shared L3 cache**
  - 96 MB, 8-way associativity

- **Off chip L4 cache**
  - 128 MB, 16-way set-associative
Memory Hierarchy: L1 Instruction cache

- Instruction Fetch Unit (IFU)
- Instruction address translation
  - 64 bit effective address -> 78 bit virtual address -> 50 bit physical address
  - 64 entry Instruction Effective to Real Address Translation (IERAT) table
  - Two levels of Data Effective to Real Address Translation (DERAT) tables
- Instruction Prefetch
  - Initiates on demand load request to L2 cache
  - Independent demand and prefetch requests for threads
  - Prefetch requests may be dropped
- Set selection
  - 32x8 Instruction Effective Address Directory
    - Set prediction for fast access
  - I-Cache Directory
    - Access in parallel to verify set selection prediction
Memory Hierarchy: L1 Data cache

- Load/Store Unit
- 128 byte line size
  - consisted of 4 sectors of 32 bytes each
  - sector based validation
- Cache store
  - Reads in-order from Store Reorder Queue
  - Sends to both L1 and L2 cache
  - But on store miss, does not allocate a line
Memory Hierarchy: L1 Data cache

- Indexed with the effective address
- Set Selection
  - Set-predict array
  - Effective address bits (51 to 56) are used to index
  - Effective address bits (32 to 50) are used as tag
- Data Prefetch
  - Sequential access detection
  - Each stream confirmation allows additional line to L1, L2 and L3
  - Excess prefetch at lower cache level to hide memory access latency
Memory Hierarchy: L2 & L3 Cache

Memory Hierarchy: L3 Cache

- 13-state coherence protocol
  1. I: Invalid
  2. ID: Deleted, do not allocate
  3. S: Shared
  4. SL: Shared, local data source
  5. T: Formerly MU, now shared
  6. TE: Formerly ME, now shared
  7. M: Modified, avoid sharing
  8. ME: Exclusive
  9. MU: Modified, bias toward sharing
  10. IG: Invalid, cached scope-state
  11. IN: Invalid, scope predictor
  12. TN: Formerly MU, now shared
  13. TEN: Formerly ME, now shared
Memory Hierarchy: L4 Cache

- Memory Subsystem
- L4 buffer cache
  - Off chip cache
  - 128 MB 16-way set-associative cache per chip
  - Data in eDRAM and directory in SRAM
Memory Hierarchy: L4 Cache

- Buffer for the main memory
  - Reduced write latency
  - Efficient scheduling of writes
    - Virtual Write Queue
      - All write request is installed to L4 cache regardless of hit/miss
      - Cache cleaner
    - Extended prefetching
      - High-confidence stream prefetch request
        - Fetches next N pairs of lines into L4 cache
      - Regular prefetch request
        - Fetches two 128 bytes lines of data from DRAM
Multiprocessors: Overview

Multiprocessors: SMP Interconnect

- On-chip SMP Interconnect
  - Eight 16B buses, split into segments
    - four flow left to right
    - pipeline the interconnect
    - propagation delay across the chip
  - Adaptive-control mechanism
    - independently controlled core frequencies
    - changes rate commands are issued
Multiprocessors: SMP Interconnect

Multiprocessors: SMP Interconnect

- Off-chip SMP Interconnect
  - Multi-tiered fully-connected topology
    - single chip
    - 4-chip group
    - 4 groups, each chip connects to the corresponding
    - up to 2 hops between any two chips
    - a maximum 192-way SMP
Multiprocessors: SMP Interconnect

- Coherence
  - Coherence scopes
    - snooping memory-coherence protocol
    - chip, group and system
  - Coherence filtering
    - limit coherence commands
    - with chip or group scope
      - bandwidth and latency
      - Problem: memory coherence across the complete system
Multiprocessors: SMP Interconnect

- Coherence
  - Coherence filtering
    - Hardware check
      - current scope v.s. relevant caches
      - MDI (Memory-Domain Indicator) for each line
        - if off the scope, mark “remote” and retry
        - recovery mechanism to reset MDI
    - Hardware prediction
      - initial scope with coherence
Thread-Level Parallelism: Transactional Lock Elision

● execute critical sections concurrently
  ○ significant improvement when conflicts are rare

● place functions into transactions
  ○ no occupation of lock in transaction
  ○ TLE failure counter set to 0 initially
  ○ read mutex lock (failure handler)
  ○ transaction proceed (failure handler)
  ○ failure counter++, revert thread state
    ■ directly falls back to locking if
      ● busy lock
      ● TLE failure counter threshold
        ○ potential wasted cycles
Thread-Level Parallelism: Thread-Level Speculation

- execute serial operations concurrently (e.g. iterations in loop)
- each operation in a transaction
  - runtime mechanism for data inter-dependencies
  - roll back and re-execute
  - commit in original serial order
    - shared control variable
    - after committing, update with identification
    - not commit without preceding identification
References

Q&A