EV8: The Post-Ultimate Alpha

Dr. Joel Emer
Intel Fellow
Intel Architecture Group
Intel Corporation
Alpha Microprocessor Overview

Higher Performance

Lower Cost

1998 1999 2000 2001 2002 2003

First System Ship
Goals

- Leadership single stream performance

- Extra multistream performance with multithreading
  - Without major architectural changes
  - Without significant additional cost
EV8 Architecture Overview

- Aggressive instruction fetch unit
- 8-wide super-scalar execution unit
- 4-way simultaneous multithreading (SMT)
- Large on-chip L2 cache
- Direct RAMBUS interface
- On-chip router for system interconnect
  - for glueless, directory-based, ccNUMA
  - with up to 512-way multiprocessing
Instruction Issue

Reduced function unit utilization due to dependencies
Superscalar Issue

Superscalar leads to more performance, but lower utilization
Predicated Issue

Time

Adds to function unit utilization, but results are thrown away
Chip Multiprocessor

Limited utilization when only running one thread
Fine Grained Multithreading

Intra-thread dependencies still limit performance
Simultaneous Multithreading

Maximum utilization of function units by independent operations
Basic Out-of-order Pipeline

Fetch → Decode/Map → Queue → Reg Read → Execute → Dcache/Store Buffer → Reg Write → Retire

Thread-blind
Architectural Abstraction

- 1 CPU with 4 Thread Processing Units (TPUs)
- Shared hardware resources
Key Design Principles

- High throughput single stream design
- Enhancements for SMT
Little’s Law

Throughput (T) = \frac{\text{Average Number of Tasks in Region (N)}}{\text{Average Latency in Region (L)}}
Little’s Law for Instruction Fetch

- \( L = \text{fixed pipe length} + \text{average memory latency} \)

- \( N = \text{number of instructions fetched} \)

\[
T = \frac{N}{L}
\]
Instruction Fetch Unit

- **Wider fetch**
  - Fetch more statically consecutive instructions
  - Limited by “trace” length

- **Trace Cache**
  - Build sequences of dynamically consecutive instructions
  - Significantly greater complexity

- **Double fetch**
  - Fetch two non-consecutive blocks of instructions
Instruction Fetch Unit

- Address Latches
- Line Predictor
- Icache
- Rate Matching Buffer
- Branch Predictor
- Jump Address Predictor
- Return Predictor
- Misprediction Calculation
- Collapsing Logic
Instruction Fetch Characteristics

- Two 8-instruction fetches per cycle
- 16 branch predictions per cycle
- Jump target prediction
- Return address prediction
- Rate matching buffer of fetched instructions
- Collapse fetched instructions into groups of 8
Execution Unit Characteristics

- Single issue queue
  - 8-wide
  - 112+ entries
- Register file
  - 512 registers
  - 16 read/8 write ports
- Function units
  - 8 integer ALUs
  - 4 floating ALUs
  - 4 memory operations (2 read/2 write)
Little’s Law for Execution Unit

- $L_{(min)} = \text{Number of cycles in pipe}$
- $T_{(desired)} = \text{Number of desired instructions per cycle (8)}$

\[
N \quad 8 \quad = \quad \frac{N}{13}
\]
Little’s Law for Execution Unit

![Diagram showing Little’s Law for the IQ](image)

- **Max IPC**
- **Average Q chunk Lifetime**

The graph illustrates the relationship between the maximum instruction per cycle (IPC) and the average Q chunk lifetime, highlighting how performance metrics evolve over time or under varying conditions.
Key Design Principles

- High throughput single stream design

- Enhancements for SMT
Additions for SMT

- Replication required resources
  - Program counters
  - Register File (architectural space)
  - Register maps
  - ...

- Sharable resources
  - Register file (rename space)
  - Instruction queue
  - Branch predictor
  - First and second level caches
  - Translation buffers
  - ....
Approaches

- Replicated resources used for…
  - all per TPU state (except register file)
  - some sharable resources where design is easier (*)
    - E.g., return stack predictor

- Shared resources used for…
  - register file (*)
  - all other sharable resources (*)

* Policy may be needed to make priority decisions
Choosing Policy
Choosing policies

- FIFO – trivial
- Round robin – easy
- Proportional – special case
- Icount-style – fair
Icount Choosing Policy
Why Does It Count Make Sense?

\[
\frac{N}{T} = \frac{L}{L}
\]

\[
\frac{N}{4} = \frac{T}{4} = \frac{L}{L}
\]
Choosers

- Address Latches
- Line Predictor
- Retire
Choosers - Fetch

Address Latches

Line Predictor

Retire
Choosers – Fetch

Address Latches

Line Predictor

Retire
Choosers - Map
Choosers - Retire

Address Latches

Line Predictor

Retire
Choosers – LD/ST numbers
Choosers – LD/ST numbers
Choosers – Miss/Store

Address Latches

Line Predictor

cache

Retire
Choosers

- Fetch Chooser - Icount
- Map Chooser - Icount
- LD/ST Number Chooser - Proportional
- Retire Chooser – Round Robin
- Load miss Chooser – Round Robin
- Store Buffer Chooser - FIFO
Area Cost of SMT Support

Uniprocessor 4-Way Core
SMT physical overhead

Total overhead: 6%±
Multiprogrammed workload

![Bar chart showing multiprogrammed workload for different types of workloads: SpecInt, SpecFP, and Mixed Int/FP. The chart compares the performance at different levels of multiprogramming (0%, 50%, 100%, 150%, 200%, 250%) for each workload category. The performance is represented by bars for 1T, 2T, 3T, and 4T.]
Decomposed SPEC95 Applications

Turb3d Swm256 Tomcatv

1T 2T 3T 4T
Multithreaded Applications

- Barnes
- Chess
- Sort
- TP

Comparison of performance across different thread counts:
- 1T
- 2T
- 4T

Performance metrics:
- Barnes: 100%
- Chess: 150%
- Sort: 300%
- TP: 250%
Acknowledgements

- Tryggve Fossum
  - Chuan-Hua Chang
  - George Chrysos
  - Steve Felix
  - Chris Gianos
  - Partha Kundu
  - Jud Leonard
  - Matt Mattina
  - Matt Reilly