Can Mainstream Processors Support Hardware Transactional Memory?

Brown University CS IPP Symposium

April 30, 2009

Dave Christie
AMD Research & Advanced Development Lab

AMD
The future is fusion

Agenda

The Disconnect

Implementation Hurdles

An Overview of ASF 2

Thoughts on Compatibility

Goal: Educate & Set Expectations

Where I'm Coming From

The driving force for ASF was not TM, but rather "the next step beyond DCAS"

- general Transactional Memory is three or four (or 25? ☺) steps beyond DCAS

Goal: Provide a mechanism for updating some smallish number of disjoint memory locations, with a fair degree of functional flexibility for software

- rather than a handful of fixed-function instructions

Disclaimer: All of this is research, *not committed to AMD product roadmap!*

The Disconnect – Academia vs. Industry

Academia has little experience in high volume, high performance commercial microprocessor development

- Very complex => large design and verification effort
- Final product must be functional & on time
- Much pressure to leverage existing designs
  - new ground-up designs are increasingly rare
- Not suitable vehicles for experimentation
  - complex widgets must have broad benefits

➔ Academia has little insight into just how constrained the opportunities in this environment really are

Where Has HTM Appeared To-Date?

Azul Systems

- Specific market, software target
- Closed environment (control over software stack)
- Arguably not a general-purpose implementation

Sun Rock

- Well, has sort of appeared
  – Production TBD? (Not speaking for Sun!)
- very constrained, opportunistic implementation
  – but still useful

The Hurdles
(well, some of them...)

Capacity

Longevity

Contention detection & management

Idiot-proofing

End-case behavior

Implementation artifacts as architecture

Contention Detection Granularity

Unit of cache coherency (cache line)

- Period
- End of discussion

Just too handy to not leverage

Anything else is just too intrusive

Capacity

How many different addresses can be monitored for contention? And at what granularity?

How many different addresses can be stored to?

How many individual stores can be done speculatively?

Intermixed transactional & non-transactional stores?

Minimum capacity guarantees?

Capacity Constraints

Memory access path is critical, and complex

Much pressure to leverage existing structures & not introduce new ones

- E.g. do not load address lines with parallel cache

The good news: speculative-execution processors have the right sort of structures

- speculative load/store queues
- relatively deep store buffers with data forwarding
- data cache

Realistic Capacities Without New Widgets

Data cache makes a nice read buffer

- built-in contention monitoring
- trades off large potential capacity for small guaranteed capacity
- read set has to fit in cache → associativity limit

Store queue makes a nice transactional write buffer

- built-in contention monitoring
- speculative execution abort capability
- write set capacity no where near dcache capacity
- naturally makes *all* stores transactional

Capacity With New Widgets

Parallel dcache a la Herlihey & Moss

- Intrusive on main load/store path

Use buffer to capture protected addresses, do monitoring

- Extends cache associativity limit at cost of lower total capacity
- Can back up original data, allowing eager versioning

Victim cache for monitored lines

- extends associativity limit; out of main LS path

Selective store buffering – extend limit on stores

➔ All add cost that needs to be justified

Transaction Longevity

Smallish read/write sets

- → shorter transaction times
- → less susceptible to interrupts
- → harder to justify context switch support

Simply abort & roll back on interrupt

System calls not supported

Nesting

Adds complexity

Many questions

- how deep (cost depends on features)
- open vs closed (closed much easier)
- abort handling – flat or nested

Easiest to flatten transaction on abort

Can see benefits to nested abort though

Contention Management

Attempted ownership of multiple cache lines → possible deadlock

Cache coherency protocol is a risky area for changes

- needs high degree of justification

Very simple to just abort on contention

- gives requester-wins semantics rather than FCFS

Timestamps?

- Add complexity, consume bus bandwidth
- not necessarily easy to add to existing protocol

Virtualization

Need transactional region to behave the same whether running in VM or natively

- Fault on such instructions in either case if in transactional region

Rules out use of interceptible instructions

- assuming no context switch support

Generally not a problem

- most such instructions touch state that can't be managed speculatively, and are of no use in txns

But rules out use Timestamp Counter

- or at least makes it susceptible to Hypervisor policy

Memory Ordering

Desirable semantics:

- strong ordering between different transactions
- strong ordering between accesses prior to txn, within txn, and following txn
  - at least, likely easier for hdw to implement, but maybe at some cost in performance
- ordering between transactional and non-transactional stores is generally a don't-care
  - private vs shared data
  - on x86, can be controlled with fence instructions

An Example Endcase Quirk

Misaligned store-to-load fowarding

- Store data normally forwarded from store buffer to subsequent load
- x86 supports misaligned accesses
- Practical forwarding mechanism only handles certain cases of misalignment
- Harder cases punted → wait for store to commit, read from cache

Stores held back in store queue? ➔ deadlock

Solution: disallow such cases by faulting or aborting

- some taken-for-granted x86 generality is lost

A Quick Overview of ASF

New Instructions

SPECULATE → start speculative region

LOCK MOV → transactional load/store

LOCK PREFETCH[W] → monitor location for probes

RELEASE → remove address from *read* set (hint)

COMMIT → make buffered updates visible,
end speculative region

ABORT → discard buffered updates,
end speculative region

Plus small addition to exception mechanism

Overall Behavior

Signify start of Speculative Region with SPECULATE

- Tells hardware where to roll back to (EIP++)

Indicate protected addresses via LOCK MOV

- And access memory as well

Optionally use RELEASE to modify read set

- Can't release a pending transactional store
- Decreases odds of overflowing dcache capacity
- Useful for walking linked list

Signify end of Speculative Region with COMMIT or ABORT

DCAS Example

```
DCAS:
        MOV       R8, RAX
        MOV       R9, RBX
retry:
        SPECULATE                  ; Speculative region begins
        JNZ       retry            ; Simplistic handling of contention,
                                   ;  interrupt or page fault
        MOV       RCX, 1           ; Tentative failure indication
        LOCK MOV RAX, [mem1]       ; Read and protect current values
        LOCK MOV RBX, [mem2]
        CMP       R8, RAX          ; Compare with expected values
        JNZ       out              ; Skip update if miscompare
        CMP       R9, RBX
        JNZ       out
        LOCK MOV [mem1], RDI       ; Update protected memory
        LOCK MOV [mem2], RSI
        XOR       RCX, RCX         ; success indication
out:
        COMMIT                     ; end of speculative region

        ; can't get here with RCX == 0 unless operation succeeded
```

Aborting

Abort on contention, capacity limit, exception, interrupt,…

EIP rolled back to instruction after SPECULATE

Stack pointer rolled back

Registers may or may not be rolled back

EAX written with abort status code

- signifies reason, retriability
  - contention, interrupt: retriable
  - capacity: not retriable
- indicates nesting level

SPECULATE++ = conditional jump on EAX non-zero

- SPECULATE clears EAX

Atomicity Guarantees

If speculative region cannot be completely and atomically executed for any reason, it is aborted

Atomicity is with respect to all external observation, not just ASF-controlled accesses

- intermediate state not observable, period (strong isolation)

Execution atomicity guaranteed by abort on interrupt

Nesting

Very simplistic

Increment nesting level on inner SPECULATE

Decrement on inner COMMIT

- no protected stores committed

Abort to outermost level

➔ Probably not particularly useful as specified, but extremely cheap to implement

Capacity

Minimum guaranteed capacity == 4

- Intentionally low to incite discussion ☺
- Actually a fairly thorny issue – makes designers nervous
- Larger potential capacity is reference-pattern dependent
  - →Requires Plan B (STM, global lock, …)

Minimum guarantee – what does it mean?

- In absense of spurious conditions, success is guaranteed – *eventually*
  - – on first try the vast majority of time

Debug Support

Two modes described in ASF specification

- Abort, then take debug trap
- Defer trap until after Commit or Abort

Third option possible: allow single step

- only useful when debugger has frozen other threads

Other traps/faults/exceptions in speculative region cause abort, then reported via standard exception mechanism

- Rolled-back instruction pointer reported on stack
- Faulting instruction pointer captured in special register; flag set in EFLAGS image on stack

Compatibility

Popular question

No insight into competitor's plans

AMD strongly believes in benefits of cross-vendor compatibility to software development community

Language standards obviously needed

In Conclusion…

So, can mainstream processors support HTM? It depends…

- Software expectations need to be realistic
- It has to be pretty simple hardware
- Tough to find the most effective tradeoffs
- Hard to avoid implementation as architecture
- Tricky to do in an evolutionary manner
- Clearly requires cooperative hdw/sfw effort

In general, current ASF errs on the side of hdw simplicity

→ Discussion, experiments encouraged

→ Definition can change

ASF Specification Availability

http://developer.amd.com/cpu/ASF/Pages/default.aspx

Feedback to ASF_Feedback@amd.com

Upcoming blog discussions at developer.amd.com

System simulator coming soon

- PTLSim based
- Reasonable timing model
- Various options beyond what's in ASF spec

Credits:

- AMD Research & Advanced Development Lab
- AMD Operating System Research Center (Dresden)

Trademark Attribution

The future is fusion