Evaluation of
AMD's Advanced Synchronization Facility
Within a Complete Transactional Memory Stack

Dave Christie
Jae-Woong Chung
Stephan Diestelhorst
Michael Hohmuth
Martin Pohlack

Christof Fetzer
Martin Nowack
Torvald Riegel

Pascal Felber
Patrick Marlier
Etienne Riviere

Advanced Micro Devices, Inc.
Technische Universität Dresden
Université de Neuchâtel
Transactional Memory (TM)

- Multi-core everywhere, need parallel software
- Often, parallel threads need to synchronize over shared memory

- Current synchronization mechanisms (locks, …) not really suitable
  - Every programmer and every program is affected → too difficult
  - Locks: deadlocks, relies on conventions, not composable
Transactional Memory (TM)

- Multi-core everywhere, need parallel software
- Often, parallel threads need to synchronize over shared memory

- Current synchronization mechanisms (locks, ...) not really suitable
  - Every programmer and every program is affected → too difficult
  - Locks: deadlocks, relies on conventions, not composable

- **TM**: programmer declares, generic TM runtime system implements
  - C++: ```__transaction { x = map.remove(key); x.refCount--; }```
  - Compiler transforms code so that it uses TM for memory accesses
  - TM runtime is a software/hardware/hybrid implementation (STM/HTM/HyTM)
Our contribution:
Realistic HTM in a realistic TM stack

- What can we expect from first-generation hardware TM (HTM) support?
- Properties of current systems shape first-gen HTM support!
Our contribution: Realistic HTM in a realistic TM stack

- What can we expect from first-generation hardware TM (HTM) support?
- Properties of current systems shape first-gen HTM support!

- Realistic HW support: AMD's Advanced Synchronization Facility (ASF)
  - x86_64 extension for lock-free programming and TM
  - Designed to be feasible to implement in high-volume microprocessors
Our contribution: Realistic HTM in a realistic TM stack

- What can we expect from first-generation hardware TM (HTM) support?
- Properties of current systems shape first-gen HTM support!

- Realistic HW support: AMD's Advanced Synchronization Facility (ASF)
  - x86_64 extension for lock-free programming and TM
  - Designed to be feasible to implement in high-volume microprocessors

- Realistic TM-aware system stack:
  - C/C++ transaction statements (__transaction{}
  - Dresden TM compiler, based on gcc + LLVM
  - Generic TM library interface (ABI)
  - TM library implemented using ASF

- Evaluation in near-cycle-accurate simulator
  - Models x86 with ASF at a high level of detail

<table>
<thead>
<tr>
<th>Program with __transaction{}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler</td>
</tr>
<tr>
<td>TM lib ABI</td>
</tr>
<tr>
<td>ASF-TM</td>
</tr>
<tr>
<td>OS</td>
</tr>
<tr>
<td>ASF</td>
</tr>
<tr>
<td>Simulator</td>
</tr>
</tbody>
</table>
Advanced Synchronization Facility (ASF)

- Proposal: Not announced for future products

- ASF provides Speculative Regions (SRs)
  - Similar to transactions: SPECULATE, COMMIT
  - Speculative (LOCK MOV) and nonspeculative loads/stores allowed (selective annotation)

DCAS:
```assembly
MOV R8, RAX
MOV R9, RBX
retry:
    SPECULATE
    JNZ retry
    LOCK MOV R10, [mem1]
    LOCK MOV RBX, [mem2]
    CMP R8, R10
    JNZ out
    CMP R9, RBX
    JNZ out
    LOCK MOV [mem1], RDI
    LOCK MOV [mem2], RSI
    XOR RCX, RCX
out:
    COMMIT
    MOV RAX, R10
```
Advanced Synchronization Facility (ASF)

- Proposal: Not announced for future products

ASF provides **Speculative Regions (SRs)**
  - Similar to transactions: SPECULATE, COMMIT
  - Speculative (LOCK MOV) and nonspeculative loads/stores allowed (selective annotation)
  - Speculative access
    → ASF monitors cacheline (R/W, W/W conflicts)
  - SR aborts on conflicts, exceeded capacity, far jumps, disallowed instructions

**DCAS:**

```assembly
MOV R8, RAX
MOV R9, RBX
retry:
  SPECULATE
  JNZ retry
  MOV RCX, 1
  LOCK MOV R10, [mem1]
  LOCK MOV RBX, [mem2]
  CMP R8, R10
  JNZ out
  CMP R9, RBX
  JNZ out
  LOCK MOV [mem1], RDI
  LOCK MOV [mem2], RSI
  XOR RCX, RCX
out:
  COMMIT
  MOV RAX, R10
```
Advanced Synchronization Facility (ASF)

- Proposal: Not announced for future products

- ASF provides **Speculative Regions (SRs)**
  - Similar to transactions: SPECULATE, COMMIT
  - Speculative (LOCK MOV) and nonspeculative loads/stores allowed (selective annotation)
  - Speculative access
    → ASF monitors cacheline (R/W, W/W conflicts)
  - SR aborts on conflicts, exceeded capacity, far jumps, disallowed instructions

  - Simple guarantees:
    - Minimal capacity
    - SR will eventually commit (unless contention / exceeded capacity / far jumps / disallowed)

```c
DCAS:
  MOV R8, RAX
  MOV R9, RBX
ret(%r8)
  SPECULATE
  JNZ retry
  LOCK MOV R10, [mem1]
  LOCK MOV RBX, [mem2]
  CMP R8, R10
  JNZ out
  CMP R9, RBX
  JNZ out
  LOCK MOV [mem1], RDI
  LOCK MOV [mem2], RSI
  XOR RCX, RCX
out:
  COMMIT
  MOV RAX, R10
```
Constraints for ASF's design

- **Realities** in the development of high-volume microprocessors
  - Costs of chip area and verification
  - Only incremental changes feasible
  - Existing CPUs are complex: out-of-order execution, …
  - HTM touches many sensitive areas
  - All corner cases have to be handled
  - Backward/forward compatibility (code, architecture)
Constraints for ASF's design

- **Realities** in the development of high-volume microprocessors
  - Costs of chip area and verification
  - Only incremental changes feasible
  - Existing CPUs are complex: out-of-order execution, …
  - HTM touches many sensitive areas
  - All corner cases have to be handled
  - Backward/forward compatibility (code, architecture)

- **High-level ASF design constraints**
  - No change to cache-coherence protocol
  - No transaction virtualization
  - Don't change behaviour of nonspeculative code (e.g., loads/stores)
  - Keep instruction set additions small
  - Keep cost of first-generation TM extensions small
  - Enable further use cases (minimal capacity for lock-free programming)
ASF Implementations

- Dedicated storage: Locked Line Buffer (LLB)
- Augmented cache: Speculative load/store bits for cachelines
- Different capacity limitations:
  - LLB size vs. cache size/associativity

- Our study:
  - LLB, optionally use L1 cache for loads
    → LLB-8, LLB-256
    → LLB-8 w/ L1, LLB-256 w/ L1

- Providing ASF's guarantees is nontrivial
  - Capacity: mispredicted branches leading to additional loads
  - Progress: Pagefaults abort SRs, but OS should see pagefaults
### Dresden TM Compiler (DTMC)

- Compiler instruments only accesses to shared memory
  - Exploits ASF's selective annotation (no capacity wasted for stack)
- Generic TM ABI important
  - Allows cross-vendor compatibility + dynamic linking
- Compiler uses link-time optimization (LTO)
  - Can do whole-program analysis/transformation/optimization

#### Source code
```c
extern long cntr;
void increment() {
  __transaction {
    cntr = cntr + 5;
  }
}
```

#### Transformed to use TM ABI
```c
extern long cntr;
void increment() {
  _ITM_beginTransaction(...);
  long l_cntr = _ITM_R8(&cntr);
  l_cntr = l_cntr + 5;
  _ITM_W8(&cntr, l_cntr);
  _ITM_commitTransaction();
}
```

#### Binary after LTO
```c
SPECULATE
JNZ handle_abort
LOCK MOV RCX, [mem1]
ADD RCX, 5
LOCK MOV [mem1], RCX
COMMIT
```
ASF-TM

- TM runtime library
  - Uses either ASF or simple software fallback (serial execution)

- Some TM functions need software aids
  - Begin: ASF SPECULATE + software setjmp + support for nesting, serial
  - Commit: ASF COMMIT + support for nesting, serial
  - Load/store functions: just use ASF’s speculative accesses
ASF-TM

- TM runtime library
  - Uses either ASF or simple software fallback (serial execution)

- Some TM functions need software aids
  - Begin: ASF SPECULATE + software setjmp + support for nesting, serial
  - Commit: ASF COMMIT + support for nesting, serial
  - Load/store functions: just use ASF’s speculative accesses

- Calling libraries: Memory allocator as an example
  - Asynchronous aborts of SRs: Can't use malloc/free/… as is even though they're thread-safe
  - Currently we use custom pre-allocation
  - We could as well let the compiler instrument malloc with ASF-TM
Evaluation: Simulator

- PTLSim with ASF extensions
  - Highly detailed x86_64 simulation (out-of-order execution, ...)
  - Close to AMD and Intel architectures
- Single-socket 2.1GHz 8-core simulated
- All experiments performed in the simulator
- Microbenchmarks, STAMP benchmark suite
  (use `__transaction{}`, compiled with DTMC)
Microbenchmarks: Scalability

Intset: LinkList
(range=512, 20% upd.)

Intset: SkipList
(range=8192, 20% upd.)

Intset: RBTree
(range=8192, 20% upd.)
Microbenchmarks: Scalability

Intset: LinkList
(range=512, 20% upd.)

Intset: SkipList
(range=8192, 20% upd.)

Better (txn/us)

Number of threads
Microbenchmarks: Scalability

**Intset: LinkList**  
(range=512, 20% upd.)

**Intset: SkipList**  
(range=8192, 20% upd.)

**Intset: RBTree**  
(range=8192, 20% upd.)
Microbenchmarks: Scalability

**Intset: LinkList**
(range=512, 20% upd.)

**Intset: SkipList**
(range=8192, 20% upd.)

**Intset: RBTree**
(range=8192, 20% upd.)

Better (txn/us) vs. Number of threads

LLB-8
LLB-256
LLB-8 w/ L1
LLB-256 w/ L1
STAMP Benchmark Suite: KMeans

- Scales well, very few aborts
- STM is pretty good compared to sequential

![Graph 1](image1.png)

![Graph 2](image2.png)

Legend:
- LLB-8
- LLB-8 w/ L1
- LLB-256
- LLB-256 w/ L1
- STM
- Sequential
- Contention
- Abort (malloc)
- Page fault
- System call
- Capacity
STAMP Benchmark Suite: Genome

- STM overhead is much larger
- LLB-8 has not enough capacity
- LLB-8/256+L1 also have capacity aborts (cache associativity)
STAMP Benchmark Suite: Vacation

- LLB-8(+L1): more than 8 modified cachelines per transaction
- LLB-256+L1 scalability limited by cache associativity
Conclusions

- ASF is a proposal by industry for realistic first-gen HW TM support
- Often sufficient to get good TM performance
- Lots of systems work on higher layers (TM library, compiler, ...)
- Full-stack TM research necessary to build ready-to-use TM systems
- Open source releases: PTLSim-ASF, Dresden TM Compiler

http://amd64.org/research/multi-and-manycore-systems.html
http://tm.inf.tu-dresden.de
http://tmware.org
http://velox-project.eu