# Intel TSX

## “Transactional Synchronization eXtensions”

## a.k.a Transactional memory

Jim Moore

Monitor team meeting 04/09/2013

# TSX Objective

- Allow concurrency in critical regions to be exploited dynamically by hardware
- Allow simpler “coarse grain” software locking model achieve “fine grain” concurrency
- Target audience is application programmers, but available at all CPLs
- ➔ TSX is available on Haswell and future Intel processors

# Terms

- Transactional Region – programmer specified critical code region
- Transactional Execution – LCPU behavior while running transaction code
- Atomic Commit – Operation making state changes simultaneous visible to all LCPUS
- Transactional Abort – Operation rolling back state changes upon failed commit
- Tx – over-loaded abbreviation for transaction

# TSX model

- LCPU enters transaction execution mode
  - LCPU architectural state check-pointed
  - Begin speculative execution not visible to other LCPUs.
- Exit with atomic commit
  - All memory changes “published” to all LCPUs
- Exit by aborting transaction
  - Memory changes abandoned
  - LCPU state rolled back to checkpoint

# TSX Model (cont)

- Multiple LCPUs may enter tx region optimistically assuming no conflicts.
- All LCPUs will commit successfully if no conflict detected.
- Or all LCPUs will abort their tx if conflict is detected.

# TSX Interfaces

- RTM “Restricted Transactional Memory”
  - General instruction access into TSX
  - Introduced with Haswell; #UD on older CPUs
- HLE “Hardware Lock Elision”
  - Backward compatible HW wrapper
  - Based on expanded prefix definitions

# RTM

- New instructions
  - xbegin – enter transactional execution
    - xbegin <iprel fallback-address>
  - xend – commit state changes; leave tx mode
  - xabort – abandon changes; leave tx mode
    - xabort <imm8 fallback-arg>
- Also available
  - xtest – report tx mode in z flag
- New RTM cpuid bit
  - Exposed starting with prod2014

# RTM example

```
#define TX_IN_REGION (-1)  // not an abort value

xbegin:    mov   $TX_IN_REGION, %eax
           xbegin   fallback
fallback:  ret
```

Similar to setjmp() semantics
Abort behaves as longjmp() time machine

# RTM example

```
Atomic_Inc(Atomic_uint32 *var) {
   int txStat;
   do {
      txStat = xbegin();                    // enter tx region
      if (txStat == TX_IN_REGION) {
         var->value++;                      // state to change
         xend();                            // attempt commit
         return;                            // woo hoo!!
      }
   } while (txStat & TX_RETRY) == TX_RETRY));
   Atomic_Inc_Fallback(var);                // left to the reader
}
```

# RTM example

- No guarantee tx will complete
- Programmer must supply fallback code path
- Implies 2 different exclusion algorithms needed – tx and non-tx

# HLE

- Based on instruction prefix hints
  - 0xf2 xacquire - elide write & enter tx mode
    - Use with HLE enabled mem writes
  - 0xf3 xrelease – commit (similar to xend)
  - repnz, repz which are ignored on older CPUs
- Also available
  - xtest (not backward compatible)
- → New HLE cpuid bit
  - → Exposed starting with prod2014

# HLE example

```
void
Atomic_Inc(Atomic_uint32 *var)
{
   Xaquire_Lock(&Atomic_Lock);   // global atomic lock
   var->value += 1;
   Xrelease_Lock(&Atomic_lock);
}
```

HLE can use standard mutex locking model!
Ross added to MX_Lock() prod2014

# HLE example

- HW elides write, stores <addr,value> into internal state
  - IsLocked() == TRUE on this CPU
  - IsLocked() == FALSE to external LCPUs
    - Allows concurrency
- Fallback path restarts region outside tx mode
  - Be careful of xtest
- Easier to use than RTM, but less flexible

# TSX details

- LCPU architectural state can be check-pointed and rolled back. (registers)
- Memory writes confined to L1 cache
- Dirty cache lines accumulated as “write set”
- Read cache lines accumulated as “read set”
- Tx execution continues until successful commit or abort

# TSX details

- Commit allows all dirty cache lines to be seen atomically by other logical CPUs.
  - Abandons read set and write set
  - Abandons check-pointed state
- Abort marks dirty cache lines as invalid
  - Abandons read set and write set
  - Rolls back to check-pointed state
    - RTM changes <ip,eax>

# TSX aborts

- Lots of things cause aborts
- Data conflicts detected because synchronization is really needed.
  - External reads of the write set
  - External writes to write set or read set
- HLE finesses this by placing lock object into read set

# TSX aborts

- Other aborts
  - False sharing (conflicts) inside cache line (64b)
  - L1 cache size or associativity limits
  - Only available for write-back cache regions
  - Pause, cpuid, xabort instructions
  - Many other control instructions “may” abort
    - vmx, movcr, syscall, intX, change segment reg,
    - Change intr flag, etc. (think context switch)
  - Interrupts, faults, traps

# TSX details

- A/D bits set from page table walks may be exported (implementation dependent).
- Faults from speculative execution are suppressed!
  - Fallback path must be provided to make progress

# RTM example

```
Atomic_Inc(Atomic_uint32 *var) {      // var->(unmapped)
   int txStat;
   do {
      txStat = xbegin();                    // enter tx region
      if (txStat == TX_IN_REGION) {
         var->value++;              // ABORT tx, suppress #PF
         xend();                    // won't get here
        return;
      }
   } while (txStat & TX_RETRY) == TX_RETRY));
   Atomic_Inc_Fallback(var); // You did test this, right?
}
```

# HLE example

Snippet from frobos 841842-tx that was failing in BT mode. xtest() resulted in separate tx and non-tx paths.

```
inTx = FALSE;
for (i = 0; i < 20; i++) {
   asm(".byte 0xf2,0xf0; add %%eax, (%%ebx)\n"
         : : "a"(eax), "b"(&ebx) : "memory");

   If (xtest()) inTx = TRUE;       /*  FIX inTx |= xtest(); */

   asm(".byte 0xf3,0xf0; sub %%eax, (%%ebx)\n"
          : : "a"(eax), "b"(&ebx) : "memory");
}
EXPECT_TRUE(inTx);
```

# Tuning – avoiding aborts

- False sharing – align to cache lines
- Global stat counters in locks – make per CPU
- Avoid rewriting unchanged shared data
- Avoid setting/clearing interrupt flag
- Avoid self modifying code

# Nesting/Recursion

- Nesting permitted
  - #levels implementation specific
  - Combined into 1 tx region with outer most region fallback
- HLE recursive lock “may” abort

# Debugging

- Debugging RTM tx execution can be challenging
- Adding debug code often causes aborts
  - Prints
  - Recording stats or tracing data
  - Breakpoints
- HW debug assists for RTM fallback
  - SW can choose to rollback to xbegin

# Performance monitoring

- New bits in Event Selection MSRs
- IN_TX (bit 32)
  - Events occurring in tx regions
- IN_TXCP (bit 33)
  - Events occurring in completed tx regions
  - Excludes events if tx aborted.
- Supported in Intel Parallel Studio XE 2013

# TSX and the monitor

- Supported beginning with prod2014
  - decoder/interpreter support
  - CPUID bits exposed to guests
- MX_Lock/MX_Unlock enhanced with HLE
- Interpreted tx regions always fallback
- BT RTM regions always fallback

# References (links)

- Exploring Intel TSX
- Coarse grained locks and TSX
- Adding Lock Elision to Linux
- Analysis of Haswell TSX
- Latest Intel SDM