Tightening Critical Section Bounds in Mixed-Criticality Systems through Preemptible Hardware Transactional Memory
Tightening Critical Section Bounds in Mixed-Criticality Systems through Preemptible Hardware Transactional Memory
Tightening Critical Section Bounds in Mixed-Criticality Systems through Preemptible Hardware Transactional Memory
Tightening Critical Section Bounds in Mixed-Criticality Systems through Preemptible Hardware Transactional Memory
Tightening Critical Section Bounds in Mixed-Criticality Systems through Preemptible Hardware Transactional Memory
Mixed-Criticality Systems

- Consolidate multiple tasks with different certification requirements into a single system
- Tasks can have multiple WCETs, reflecting their certification/validation level
- Two tasks H (1, 2, 2) and L (1, −, 2) with \( (c_{\text{Low}}, c_{\text{High}}, \text{period}) \)
Critical Sections

Low critical task

High critical task

Shared memory buffer
Critical Sections

Low critical task

High critical task

Shared memory buffer
Critical Sections

Low critical task  High critical task

Shared memory buffer
Critical Sections in MC Systems

- Low critical tasks are dropped, if the system changes into high criticality mode.
- But: shared resource cannot be revoked, thus accessing it has to be evaluated equal to the highest criticality of any task accessing it (compare: ceiling priority).
- Revocation using transactional memory.
Hardware Transactional Memory

- Accumulate changes locally, commit or abort atomically, i.e. all-or-nothing
- HTM not used for synchronization (locks are still used), but to quickly abort low-criticality tasks even if they are in a critical section
Ceiling Priority + Commit

1. acquire R
2. set timeout + raise priority
3. xbegin; R; xend
4. release R
5. stop timeout + lower priority

\( W_1(LO) \)

\( \widehat{\mathcal{R}} \)

\( \tau_2 \)

\( \tau_1 \)

user

kernel

priority

time
Ceiling Priority + Timeout

1. acquire R
2. set timeout + raise priority
3. xbegin; R...
4. timeout aborts transaction
5. lower priority

R
\hat{R}
\tau_2
\tau_1
user
kernel
time
priority
W_1(LO)
Implementation in gem5

- Cycle-accurate Out-of-Order CPU
- Added Intel-like HTM support (XBegin/XEnd)
- L1 Data cache lines with 'T'-Bit
- Transactional mode in CPU and Cache
- Adapted snoop logic to properly handle remote accesses to transactional cache lines
XPreempt / XResume

• But: Intel-like transactions are fragile, abort on various reasons, e.g. exceptions and interrupts

• This must not happen, so we added XPreempt/XResume to pause/unpause transactions
Evaluation

- Random-generated task sets (up to 10 tasks) using UUnifast algorithm
- Periods between 4 and 144, resulting in a hyperperiod of 5184
- Dual-criticality systems, where High-WCET is 1.2 / 1.5 / 2.0 times the respective Low-WCET
- System load up to 0.5 / 0.75 / 1.0 and 1.5 for multi processor setup
50% System Load

CPU load

number of tasksets

0  20  40

0.4  0.45  0.5  0.55  0.6  0.65  0.7  0.75

1.2
1.5
2.0

CPU load

number of tasksets

0  20  40

0.4  0.45  0.5  0.55  0.6  0.65  0.7  0.75

1.2
1.5
2.0

Mixed-criticality HTM
100% System Load

![Graph showing CPU load and taskset number for different criticalities](image)

- CPU load on the x-axis
- Number of tasksets on the y-axis

Criticalities: 1.2, 1.5, 2.0

Benjamin Engel

Mixed-criticality HTM
150 % System Load on SMP

**Graphs:**
- Upper graph: X-axis: CPU load, Y-axis: number of tasksets, three lines for different values: 1.2, 1.5, 2.0.
- Lower graph: Similar setup but without a legend.
## Deadline misses without HTM

### Low-crit to High-crit ratio

<table>
<thead>
<tr>
<th>System load</th>
<th>1 : 1.2</th>
<th>1 : 1.5</th>
<th>1 : 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>1%</td>
<td>6%</td>
<td>21%</td>
</tr>
<tr>
<td>75%</td>
<td>4%</td>
<td>14%</td>
<td>54%</td>
</tr>
<tr>
<td>100%</td>
<td>45%</td>
<td>99.6%</td>
<td>---</td>
</tr>
<tr>
<td>150%</td>
<td>5%</td>
<td>26%</td>
<td>50%</td>
</tr>
</tbody>
</table>
In a Nutshell

- Using transactional memory to enforce tighter critical section bounds
- Implementation of preemptible HTM in gem5
- Improved schedulability and reduced system load using HTM by allowing low criticality resource access times being trustworthy