DYNAMIC ARBITRATION OF MEMORY REQUESTS
WITH TDM-LIKE GUARANTEES

OVSTR 11/06/2018
• Multi-core architectures

• Tasks with different criticality levels

• Shared memory

• Complex analysis:
  • Many interactions among cores/tasks
  • When does contention occur?
  • How is contention resolved?

A possible solution: TDM arbitration policy
• Fixed time windows
• Exclusive access to the main memory

• Isolation between cores
• Predictable behavior

• Not work-conserving
• Low resource utilization

Our goal: improve resource utilization while keeping TDM guarantees
• System model

• Strict TDM arbitration policy

• Dynamic TDM

• Dynamic TDM with slack accumulation

• Experiments

• Current work
SYSTEM MODEL

- $m$ processing cores
- A single main memory
- A single task per core
  - Sequence of numbers representing distance (computing time) between memory accesses
  - Example: $A = (2, 2, 0)$
    - $A_0$: First access after two cycles
    - $A_1$: Second access 2 cycles after completion of first access
    - $A_2$: Third access immediately after completion of second

Completion of memory access  
Memory request

Computing time  
Memory time

A release  
A_0  
A_1  
A_2  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
OUTLINE

• System model

• Strict TDM arbitration policy

• Dynamic TDM

• Dynamic TDM with slack accumulation

• Experiments

• Current work
Critical tasks

Memory sequences
A = (2, 2, 0)
B = (42, 0)

Strict TDM arbitration
TDM slot length : 8 cycles
2 active cores → TDM period : 16 cycles
ILLUSTRATING EXAMPLE STRICT TDM

\[ A = (2, 2, 0) \quad B = (42, 0) \]

Execution length: 10 slots
Used slot: 50%
Unused slots: 50%

Slot Length,
\( SL = 8 \) cycles
TDM period,
\( TDMP = 2 \times SL = 16 \) cycles

Not work-conserving
Low resource utilization
OUTLINE

• System model

• Strict TDM arbitration policy

• Dynamic TDM

• Dynamic TDM with slack accumulation

• Experiments

• Current work
WHY DYNAMIC TDM?

• Strict TDM arbitration
  • Predictable
  • Not-work conserving (unused slots)
  • Low resource utilization

• Dynamic TDM
  • Take advantage of unused slots
  • Keep TDM guarantees
  • Every request carries a deadline
  • Deadline driven arbitration policy
  • Improve resource utilization
DYNAMIC TDM

Critical tasks
- A
- B

Non-Critical tasks
- c
- d

Memory sequences
- \( A = (2, 2, 0) \)
- \( B = (42, 0) \)
- \( c = (0, 4, 12, 2) \)
- \( d = (28, 36) \)

Dynamic TDM arbitration
- TDM slot length : 8 cycles
- 2 critical tasks \( \rightarrow \) TDM period : 16 cycles
DYNAMIC ARBITRATION SCHEME

• Critical tasks
  • Deadline = completion date under strict TDM

• Non-critical tasks
  • Dynamically updated deadline: end next TDM slot
  • Reclaim unused slots

• Earliest deadline arbitration policy
  • Critical prioritized over non-critical

Critical tasks
A=(2, 2, 0)
B=(42, 0)

Non-critical tasks
c=(0, 4, 12, 2)
d=(28, 36)

Execution length: 13 slots
Used slot: 85%
Unused slots: 15%
COMPARISON BETWEEN TDM SCHEMES

**Strict TDM**
- Execution length: 10 slots
- Used slot: 50%
- Unused slots: 50%

**Dynamic TDM**
- Execution length: 13 slots
- Used slot: 85%
- Unused slots: 15%

Critical requests complete earlier
• System model

• Strict TDM arbitration policy

• Dynamic TDM

• Dynamic TDM with slack accumulation

• Experiments

• Current work
DYNAMIC ARBITRATION SCHEME WITH SLACK COUNTER

- Track early completed requests
- Store the time won compared to Strict TDM in a Slack Counter
- Slack counter $X_\Delta(r_i) = dl(r_{i-1}) - C(r_{i-1})$
  - $C(r_{i-1})$: Completion date of request $r_{i-1}$
  - $dl(r_{i-1})$: Deadline of request $r_{i-1}$
- Non-critical tasks may be prioritized over Critical tasks

\[
X_\Delta(A_1) = dl(A_0) - C(A_0)
X_\Delta(A_1) = 24 - 16
X_\Delta(A_1) = 8
\]

Execution length: 11 slots
Used slot: 100%
Unused slots: 0%
COMPARISON BETWEEN TDM SCHEMES

**Dynamic TDM with slack:**
- TDM guarantees for critical tasks
- Improve resource utilization

**Execution length:** 11 slots
- **Used slot:** 100%
- **Unused slots:** 0%

**Strict TDM**
- Execution length: 10 slots
- Used slot: 50%
- Unused slots: 50%

**Dynamic TDM without slack**
- Execution length: 13 slots
- Used slot: 85%
- Unused slots: 15%
OUTLINE

- System model
- Strict TDM arbitration policy
- Dynamic TDM
- Dynamic TDM with slack accumulation
- Experiments
- Current work
EVALUATION: HARDWARE SETUP

- 4 Patmos cores
  - 32 KB Method cache, with LRU replacement policy
  - 256 byte stack cache
  - 32 KB Data cache with write-through and LRU replacement policy
- Main memory
  - Access latency = 21 cycles
  - Duration of a TDM slot = 21 cycles
EVALUATION: BENCHMARKS

MIBench benchmarks
- 4 benchmarks with longest runtime
- 2 critical: rijndael (A), blowfish (B)
- 2 non-critical: djikstra (c), adpcm (d)
- Parallel execution

Recording interesting metrics
- Number of granted memory accesses
- Average memory wait time
- Average slack counter values
CHARACTERIZATION OF THE STRONG ACTIVITY OF SLACK COUNTERS

- Number of TDM slots simulated: $\sim 2 \cdot 10^7$
- A data point represents an average value of $\sim 6400$ TDM slots
OUTLINE

• System model
• Strict TDM arbitration policy
• Dynamic TDM
• Dynamic TDM with slack accumulation
• Experiments
• Current work
• Issue date not aligned with a TDM start
• Request allocated to the next TDM slot

**Issue delay**: *waiting time between the issue date and the allocation date*
Variable memory access latency
Fixed TDM slot length
Real access latency often less than TDM slot length

**Release delay**: Difference between the end of a TDM slot and the real completion time
BENCHMARK SETUP

- Evaluation of release and issue delays
- Randomly generated synthetic tasks
- Memory traffic generator
- Simulation setup
  - Number of tasks/cores (4, 8, 12, 16, 20, 24)
  - Global system utilization (between 10% to 100%)
  - Criticality repartition
    - 25% critical and 75% non-critical
    - 50% critical and 50% non-critical
TOTAL IDLE TIME

Condensed average memory idle time from all runs

Strict TDM

Dynamic TDM with slack counter
CONCLUSION AND FUTURE WORK

Dynamic approach based on TDM with slack counters

- TDM guarantees: deadline = completion date under strict TDM
- Improved resource utilization
- Possibility to reorder slots (DDR control schedule)
- Preserves results of slot offset analysis

Preliminary evaluation

- Interesting behavior of slack counters

Investigate more realistic execution models

Implement in hardware

Other forms of slack to be accumulated
HOW TO GET A DYNAMIC ARBITRATION?

Deadlines

Non-critical tasks

\[ dl = curPeriod + So \]
\[ if (Ad - curPeriod > So) \]
\[ dl = dl + Sl \]

Critical tasks

\[ dl = curPeriod + So \]
\[ if (Ad - curPeriod > So) \]
\[ dl = dl + TDMP \]

\[ So = (slotNumber - 1) \times Sl \]

**slotOffset**,

\[ Sl : Slot length \]
\[ curPeriod : start date of the current TDM period \]
\[ TDMP : TDM period \]
WHY SLACK COUNTER?

- Arrival date of critical tasks
  - $A_v(r_i) = C(r_{i-1}) + 1 + X(i) + X_\Delta(r_i)$
  - Identical to strict TDM

- Example:
  - Request $A = (2, 2, 0)$
  - $A_v(A_2) = 31 + 1 + 0 + 8$
  - $A_v(A_2) = 40$
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Memory accesses</th>
<th>Isolated execution Time (Cycles)</th>
<th>Concurrent execution Time (Cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rijndael (A)</td>
<td>7165378</td>
<td>236881722</td>
<td>349949607</td>
</tr>
<tr>
<td>Blowfish (B)</td>
<td>3527173</td>
<td>219815862</td>
<td>237530328</td>
</tr>
<tr>
<td>Djikstra (c)</td>
<td>4856155</td>
<td>209856906</td>
<td>362402145</td>
</tr>
<tr>
<td>Adpcm (d)</td>
<td>1716629</td>
<td>141243669</td>
<td>405932856</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>17265335</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ILLUSTRATING EXAMPLE STRICT TDM

\[ A = (2, 2, 0) \quad B = (42, 0) \quad c = (0, 4, 12, 2) \quad d = (28, 36) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

**Slots Length,**
**SL = 8 cycles**

**TDM period,**
**TDMP = 4 * SL = 32 cycles**