Cache Design and Timing Analysis for Preemptive Multi-tasking Real-Time Uniprocessor Systems

PhD Defense by Yudong Tan
Advisor: Vincent J. Mooney III

Center for Research on Embedded Systems and Technology (CREST)
School of Electrical and Computer Engineering
Georgia Institute of Technology
OUTLINE

- Motivation
- Problem Statement
- Previous Work
- WCRT Analysis
- Prioritized Cache Design
- Experiments for WCRT Analysis
- Experiments for Prioritized Cache
- Conclusions
- Publications
Motivation

- Timing analysis are critical for real-time systems
  - Correctness
    - Functional
    - Timing
  - Hard real-time systems
    - Strict timing constraints
    - Robots, Mars Rover, Automobiles etc.
  - Soft real-time system
    - Less strict timing constraints
    - A tighter schedule for QoS, utilization of resources
    - Video/Audio applications etc.
  - Worst Case Timing Analysis
Motivation

- An embedded real-time system integrates hardware and software.
- Hardware has strict timing properties.
  - OS components: Lock Cache, DMMU, DDU, DAU etc.
  - Applications: MPEG encoder/decoder, network etc.
- Software is a problem for timing analysis.
  - Flexible, easy to develop and upgrade
  - Difficulties in timing analysis
    - Branches
    - Pipelining
    - Cache
    - Out-of-order execution
    - ...

Diagram:
- FFT
- IDCT
- Analog Components
- Reconfigurable Logic
- Control Processor
- DMA
- CPU
- L1 $
Motivation

- Software performance is affected by cache greatly.

- Pros
  - Reduce memory access time
  - Accelerate execution in average

- Cons
  - Memory access time is unpredictable.
  - Cache interference among tasks complicates timing analysis.

- Cache related timing analysis is needed.
**Terminology**

- **Worst Case Execution Time (WCET)**
  - The time taken by a task to complete its computations in the worst case

- **Worst Case Response Time (WCRT)**
  - The time taken by a task from its arrival to its completion of computations in the worst case

- **Cache Related Preemption Delay (CRPD)**
  - Cache reload cost caused by cache interference between the preempted and the preemting task

- **Schedulability Analysis**
  - The procedure performed to analyze if a feasible schedule exists for a particular real-time system under consideration.

\[
\text{WCET of } T_1: r_1 + r_3; \quad \text{WCRT of } T_1: r_1 + r_2 + r_3
\]

Ti,j is the jth run of Task Ti
Problem Statement

Objective

- WCRT analysis in a preemptive multi-tasking system
  - Including Cache Related Preemption Delay (CRPD)
  - Computationally efficient
  - Schedulability analysis
- Customizing cache to reduce cache interference
- WCRT analysis for the customized cache

Assumption

- Multi-tasking, Preemptive
- Uniprocessor, Unified L1 cache (Set Associative / Direct Mapped), Two-level memory hierarchy
- Fixed Priority Scheduling (e.g., RMS)
Five Major Contributions

1. A novel approach is proposed to analyze inter-task cache interference.
   ✓ Cache Index Induced Partition (CIIP)

2. Inter-task cache interference analysis is integrated with intra-task cache interference analysis.

3. Path analysis is used to improve cache interference analysis.

4. A new WCRT estimate formula is proposed.
   ✓ Polynomial complexity vs. exponential complexity of the best known prior approach (Lee’s approach)
   ✓ Infeasible preemptions removed.
   ✓ Tighter WCRT estimate

5. A novel “prioritized cache” design is presented to reduce CRPD.
   ✓ Task priority considered in cache partition allocation
   ✓ WCRT analysis is applied to the prioritized cache.
     ▪ Safer than benchmarking
OUTLINE

- Motivation
- Problem Statement
- Previous Work
- WCRT Analysis
- Prioritized Cache Design
- Experiments for WCRT Analysis
- Experiments for Prioritized Cache
- Conclusions
- Publications
Previous Work

- Cache-related static timing analysis
  - WCET analysis
  - WCRT analysis
- Cache usage customization
  - Hardware cache partitioning
  - Software cache customization
  - More predictable cache behavior
- Timing analysis for customized caches
Previous Work: WCET Analysis

- Static Analysis of Single Task Worst Case Execution Time (WCET)
  - SYMTA (SYMbolic Timing Analysis), [Wolf and Ernst]
    - Extend basic blocks to Single Feasible Path Program Segments
    - Reduce over-estimate of WCET on boundaries of basic blocks
  - Other WCET analysis approaches exist
Previous Work: WCRT Analysis

- Basic WCRT Analysis [Tindell] [Joseph & Pandya]
  - WCRT Analysis without considering cache
  - Iterative calculation

$T_i$ All tasks in the system sorted in the descending order of their priorities.

$C_i$ WCET of $T_i$  $P_i$ Period of $T_i$, also defines the deadline

$hp(i)$ The set of tasks with higher priorities than $T_i$

Response time

$$R_i^0 = C_i$$

$$R_i^k = C_i + \sum_{j \in hp(i)} \left[ \frac{R_i^{k-1}}{P_j} \right] \times C_j$$
Previous Work: WCRT Analysis

- A cache related WCRT analysis approach [Busquests-Mataix’s et al.]
- CRPD overestimated: all cache lines used by the preemting task have to be reloaded.

A direct mapped cache

overestimated

overlapped

T1,1

T2,1

T1

T2
Previous Work: WCRT Analysis

Lee’s Approach

- Best known prior approach
- Intra-task eviction analysis – CRPD related to each preemption
  - Reaching Memory Blocks (RMB)
    - all possible memory blocks that may reside in the cache when the task reaches an execution point $s$
  - Living Memory Blocks (LMB)
    - all possible memory blocks that may be one of the first $L$ distinct references to the cache after execution point $s$, where $L$ is the number of ways in the cache.
  - Useful memory blocks: memory blocks used before the preemption and requested after the preemption by the preempted task
  - Overestimate - Not all useful memory blocks need to be reloaded.
- ILP – the number of preemptions
  - May include infeasible preemptions
  - All preemption scenario considered – Exponential computational complexity

Other WCRT analysis approaches

A direct mapped cache with 16 lines, 16 bytes in each line

<table>
<thead>
<tr>
<th>Memory trace</th>
<th>Execution Point S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0010</td>
<td></td>
</tr>
<tr>
<td>0x0020</td>
<td></td>
</tr>
<tr>
<td>0x0030</td>
<td></td>
</tr>
<tr>
<td>0x0110</td>
<td></td>
</tr>
<tr>
<td>0x0020</td>
<td></td>
</tr>
<tr>
<td>0x0030</td>
<td></td>
</tr>
</tbody>
</table>

RMB={0x0020, 0x0030, 0x0110}
LMB={0x0020, 0x0030}
UMB={0x0020, 0x0030}
Previous Work: WCRT Analysis

- **Comparison with Prior Approach**
  - **Tighter CRPD estimate**
    - A formal analysis approach for inter-task cache eviction.
    - No inter-task cache eviction analysis method proposed in Lee’s approach
    - Integration of inter- and intra-task cache eviction analysis
    - Path analysis
  - **Tighter WCRT estimate**
    - No infeasible preemptions
    - Tighter CRPD
  - **Less complex in computation**
    - Polynomial vs. Exponential
Previous Work: Customize Cache Usage (1)

- **Hardware approaches**
  - SMART (Strategic Memory Allocation for Real-Time Systems) Cache [Kirk]
    - Assign cache lines to tasks according to their CPU utilization
  - Column Cache [Suh and Rudolph]
    - Cache is partitioned at the granularity of cache columns
    - Data cache only
  - Lock Cache [Maki], Data Cache Lock [Vera], Split Cache [Juan]

- **Compare the prioritized cache with prior hardware approaches**
  - Partition a cache at the granularity of columns – No need to change tag size
  - Assign cache partitions according to task priorities
  - Easy usage
    - Minor modification in the OS for transparent support
    - No specific instructions needed
  - Apply to instruction caches and data caches
  - Formal WCRT analysis
Previous Work: Customize Cache Usage (2)

- Software Approaches
  - Main idea: Manipulating memory-to-cache mapping
  - Software-based cache partitioning approach [Wolfe]
  - Customize memory-to-cache mapping [Wager]
  - OS-Controlled Cache Predictability [Liedtke]

- Compare the prioritized cache with prior software approaches
  - Do not need sophisticated modification of OS or compilers
  - Do not need to insert additional instructions to tasks
  - Do not need to control memory-to-cache mapping directly
    - ✓ No problem with pre-compiled libraries
    - ✓ No additional memory fragmentation problem
Previous Work: Timing Analysis for Customized Caches

- Average timing performance evaluation via benchmarking
  - MPEG [Dropsho]
  - GZIP [Suh & Rudolph]
  - Livermoreloop Fortran Kernels (LFK) [Maki]
- No guarantee for worst case timing performance
- Our approach: WCRT analysis for the prioritized cache
  - Provide a safe base for using a prioritized cache in a real-time system
Flow of Overall Approach

Real-time applications

Simulation Platform

WCET Of SFP-PrS

SYMTA

WCRT Analysis

Intra-task cache eviction analysis

Inter-task cache eviction analysis

Path analysis

WCRT estimate

Schedulability analysis

WCRT & Schedulability
Cache Related Preemption Delay (CRPD)

- What causes CRPD?
  - Cache interference between the preempted task and the preempting task – cache reload cost

- Why do cache lines need to be reloaded?
  - Inter-task cache eviction
  - Intra-task cache dependency
Inter-task Cache Eviction

- Two Tasks: T1 and T2
- T2 has a higher priority than T1

Condition 1
Only cache lines used by both the preempting and the preempted task possibly need to be reloaded.
Intra-task Cache Dependency

Task T1

Direct-mapped Cache
16 bytes/line, 16 lines

index
0 1 2 3 ...
F

<table>
<thead>
<tr>
<th>0x0110</th>
<th>0x0114</th>
<th>0x0118</th>
<th>0x011C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0120</td>
<td>0x0124</td>
<td>0x0128</td>
<td>0x012C</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0x00F0</td>
<td>0x00F4</td>
<td>0x00F8</td>
<td>0x00FC</td>
</tr>
</tbody>
</table>

Useful Memory Blocks

Not a useful memory block
Cache Interference Analysis by Using Intra-task Cache Dependency

Preempted task T1

Preempted task T2

Preempted task T1

Preempted (execution point s)

Preempting task T2

Condition 2

Only memory blocks that are used before the preemption and requested after the preemption by the preempted task (useful memory blocks) potentially need to be reloaded.

No need to reload all cache lines satisfying Condition 1.
Two Conditions for Cache Reload

- Used by both the preempting and the preempted task.
  - Only cache lines in the intersection set of cache lines used by the preempting task and the preempted task.
- Loaded to the cache before the preemption and requested after the preemption by the preempted task.
  - Only cache lines mapped from “useful memory blocks”.
Inter-task Cache Eviction Analysis

- Memory trace (No dynamic data allocation)
- Memory vs. Cache
  - Index of a memory block vs. cache line
  - Only memory blocks with the same index can possibly conflict in the cache.
- An example

Two Sets of Memory Blocks:

M1={0x700; 0x800; 0x710; 0x810; 0x910}
M2={0x200; 0x310; 0x410; 0x510}

A 4-way set associative cache with 16 sets, each line has 16 bytes.
**Inter-task Cache Eviction (Cont.)**

- **Cache Index Induced Partition (CIIP)**
  - Partition a memory block set according to their index
  - Memory blocks in the same partition have the same index.
  - Cache eviction can only happen among memory blocks in the same partition.

An \( L \)-way set associative cache with \( N \) lines in each set.

\[
M = \{m_0, m_1, \ldots, m_K\}
\]

CIIP of \( M \):

\[
\hat{M} = \{\hat{m}_0, \hat{m}_1, \ldots, \hat{m}_{N-1}\}
\]

Where,

\[
\hat{m}_i = \{m_j \in M \mid idx(m_j) = i\}
\]

- **An example of CIIP**

\( M_1 = \{0x700; 0x800; 0x810; 0x810; 0x910\} \)

\( \hat{m}_{10} = \{0x700, 0x800\} \) \quad \text{Index=0} \)

\( \hat{m}_{11} = \{0x710, 0x810, 0x910\} \) \quad \text{Index=1} \)

\( \hat{M}_1 = \{\hat{m}_{10}, \hat{m}_{11}\} = \{\{0x700; 0x800\}, \{0x710; 0x810; 0x910\}\} \)
Inter-task Cache Eviction Analysis (Cont.)

Use CIIP to estimate the upper bound of inter-task cache eviction cost

\[ M_1 = \{m_{10}, m_{11}, \ldots, m_{1K_1}\} \quad \hat{M}_1 = \{\hat{m}_{10}, \hat{m}_{11}, \ldots, \hat{m}_{1,N-1}\} \]

\[ M_2 = \{m_{20}, m_{21}, \ldots, m_{2K_2}\} \quad \hat{M}_2 = \{\hat{m}_{20}, \hat{m}_{21}, \ldots, \hat{m}_{2,N-1}\} \]

Upper bound on the number of memory blocks that possibly conflict in the cache:

\[ S(M_1, M_2) = \sum_{r=0}^{N-1} \min(L, |\hat{m}_{1r}|, |\hat{m}_{2r}|) \]

Complexity: Linear to the number of memory blocks

**Contribution 1:** A novel approach is proposed to analyze inter-task cache interference.
Inter-task Cache Eviction Analysis (Cont.)

An example

A 4-way SA cache with 16 sets, each line has 16 bytes.

Two Sets of Memory Blocks:

\[ M_1 = \{0x700; 0x800; 0x900; 0x910\} \quad \hat{M}_1 = \{\hat{m}_{10}, \hat{m}_{11}\} = \{0x700; 0x800\}, \{0x900; 0x910\} \]

\[ M_2 = \{0x200; 0x300; 0x400; 0x500\} \quad \hat{M}_2 = \{\hat{m}_{20}, \hat{m}_{21}\} = \{0x200\}, \{0x300; 0x400; 0x500\} \]

CASE 1

\[ \min(\hat{m}_{10}, \hat{m}_{20}, 4) = 1 \]

\[ \min(\hat{m}_{11}, \hat{m}_{21}, 4) = 3 \]

\[ S(M_1, M_2) = 1 + 3 = 4 \]

CASE 2

\[ \min(\hat{m}_{10}, \hat{m}_{20}, 4) = 3 \]

\[ \min(\hat{m}_{11}, \hat{m}_{21}, 4) = 3 \]

\[ S(M_1, M_2) = 3 + 3 = 6 \]

-- gives an upper bound
Intra-task Cache Eviction Analysis

- Useful Memory Blocks (UMB) at the execution point
  - Intersection of RMB and LMB at the execution point
  - Only useful memory blocks need to be reloaded.

- Maximum Useful Memory Block Set (MUMBS)
  - The maximum set of UMB over all the execution points of a task
Integrate Inter- and Intra-task Cache Eviction Analysis

- Only useful memory blocks are potentially required to be reloaded.
- MUMBS of the preempted task is used in the CIIP calculation.

\[
M_1 = \{m_{10}, m_{11}, \ldots, m_{1K_1}\} \quad \hat{M}_1 = \{\hat{m}_{10}, \hat{m}_{11}, \ldots, \hat{m}_{1,N-1}\}
\]

\[
M_2 = \{m_{20}, m_{21}, \ldots, m_{2K_2}\} \quad \hat{M}_2 = \{\hat{m}_{20}, \hat{m}_{21}, \ldots, \hat{m}_{2,N-1}\}
\]

MUMBS

\[
\tilde{M}_2 = \{\tilde{m}_{20}, \tilde{m}_{21}, \ldots, \tilde{m}_{2K_2}\} \quad \hat{\tilde{M}}_2 = \{\hat{\tilde{m}}_{20}, \hat{\tilde{m}}_{21}, \ldots, \hat{\tilde{m}}_{2,N-1}\}
\]

Without considering UMBs

\[
S(M_1, M_2) = \sum_{r=0}^{N-1} \min(L, |\hat{m}_{1r}|, |\hat{m}_{2r}|) \tag{1}
\]

With considering UMBs

\[
S(M_1, \tilde{M}_2) = \sum_{r=0}^{N-1} \min(L, |\hat{m}_{1r}|, |\hat{\tilde{m}}_{2r}|) \tag{2}
\]

\[
\tilde{M}_2 \subseteq M_2 \quad \Rightarrow \quad S(M_1, \tilde{M}_2) \leq S(M_1, M_2)
\]

Contribution 2: Inter-task cache interference analysis is integrated with intra-task cache interference analysis.
An Example

A direct mapped cache \((L=1)\) with 16 lines, 16 bytes in each line

\[
M_1 = \{0x1010,0x1020,0x1030\} \quad \hat{M}_1 = \{\{0x1010\},\{0x1020\},\{0x1030\}\}
\]

\[
M_2 = \{0x0010,0x0020,0x0030,0x0100,0x0110,0x0120,0x0130\}
\]

\[
\hat{M}_2 = \{\{0x0100\},\{0x0010,0x0110\},\{0x0020,0x0120\},\{0x0030,0x0130\}\}
\]

\[
UMB_0 = \{} \quad UMB_1 = \{0x0020\} \quad UMB_2 = \{} \quad UMB_3 = \{} \quad UMB_4 = \{0x0100,0x0020\} \quad UMB_5 = \{0x0100,0x0020\} \quad UMB_6 = \{0x0100\} \quad UMB_7 = \{0x0100\}
\]

\[
\hat{M}_2 = \{0x0100,0x0020\} \quad \hat{M}_2 = \{\{0x0100\},\{0x0020\}\}
\]

\[
S(M_1, M_2) = 3 \quad \text{No intra-task cache dependency considered}
\]

\[
S(M_1, \hat{M}_2) = 1
\]
Path Analysis

- A real application may have multiple feasible paths.

An example: T1 - An Edge Detection application with two algorithms.
T2 - OFDM, lower priority

Path 1: 4 lines overlapped
Path 2: 2 lines overlapped
No path analysis: 5 lines overlapped

Op_type?

Cauchy Alg
Sobel Alg

Cache lines used by OFDM
Path Analysis (Cont.)

Two tasks: \( T_i \) \( T_j \)

\( T_j \) has multiple paths, \( Pa_i^k \)

\( M_j^k \) Memory set accessed by \( T_j \) when it runs along path \( Pa_i^k \)

\( \tilde{M}_i \) The MUMBS of \( T_i \)

Cost of a path in \( T_j \)

\[
C(Pa_j^k) = S(\tilde{M}_i, M_j^k) = \sum_{r=0}^{N-1} \min(L, |\tilde{m}_{i,r}|, |\tilde{m}_{j,r}|)
\]

The problem is converted to find the worst path in \( T_j \)

Worst MUMBS Path \( Pa_j^{WMP} \) Memory set \( M_j^{WMP} \)

We search all possible paths in \( T_j \)

Contribution 3: Path analysis is used to improve cache interference analysis.
CRPD Estimate

- CRPD estimate
  - Fixed cache miss penalty
  - Two tasks $T_1$ and $T_2$, $T_2$ has a higher priority than $T_1$.
  - Memory trace $M_1$ and $M_2$

\[
CRPD(T_i, T_j) = C(Pa_j^{WMP}) \times C_{miss}
\]
Nested Preemptions

By considering nested preemptions:

\[
C(Pa_j^k) = S(\bigcup_{l=j+1}^i \tilde{M}_l, M_j^k) = \sum_{r=0}^{N-1} \min(L, \bigcup_{l=j+1}^{i} \tilde{m}_{l,r}, \tilde{m}_{j,r}^k)
\]

\[
CRPD(T_i, T_j) = C(Pa_j^{WMP}) \times C_{miss}
\]
Improved WCRT Analysis

- WCRT without CRPD
  \[ R_i^0 = C_i \]
  \[ R_i^k = C_i + \sum_{j \in hp(i)} \left( \frac{R_{i}^{k-1}}{P_j} \right) \times C_j \]

- WCRT with CRPD
  \[ R_i^0 = C_i \]
  \[ R_i^k = C_i + \sum_{j \in hp(i)} \left( \frac{R_{i}^{k-1}}{P_j} \right) \times (C_j + CRPD(T_i, T_j) + 2C_{cs}) \]

Twice Context Switch: one for preemption and one for resuming

- An example

<table>
<thead>
<tr>
<th>Task</th>
<th>WCET</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>T2</td>
<td>49</td>
<td>100</td>
</tr>
</tbody>
</table>

\[ CRPD(T_2, T_1) = 3 \]
\[ C_{cs} = 1 \]

\[ R_2^0 = 49 \quad R_2^1 = 49 + \left( \frac{49}{30} \right) \times (5 + 3 + 2) = 69 \]

\[ R_2^1 = 49 + \left( \frac{69}{30} \right) \times (5 + 3 + 2) = 79 \quad R_2^2 = 49 + \left( \frac{79}{30} \right) \times (5 + 3 + 2) = 79 \]

\[ R_2 = 79 \]
Improved WCRT Analysis (Cont.)

- WCRT estimated for each task in the descending order of priorities of tasks

- Computational Complexity
  - The number of iterations for each task is bounded by $\frac{P_i}{P_0}$
  - The computational complexity in each iteration is proportional to the number of tasks.
  - All tasks except the task with the highest priority need to be estimated.
  - The total computation complexity is $O(n^2)$, where $n$ is the number of tasks.

Contribution 4: A new WCRT estimate formula is proposed.
Improved WCRT Analysis (Cont.)

- Comparison of computation complexity
  - Lee’s approach: Exponential
    - All preemption scenarios have to be considered by using ILP.
  - Our approach: Polynomial
    - CRPD covers the nested preemptions
    - No need to explore all preemption scenarios.
Schedulability

- The tasks are schedulable if:
  - The iteration above converges.
  - The WCRT of all tasks are less than their periods.
OUTLINE

- Motivation
- Problem Statement
- Previous Work
- WCRT Analysis
- Prioritized Cache Design
- Experiments for WCRT Analysis
- Experiments for Prioritized Cache
- Conclusions
- Publications
Prioritized Cache

Motivation
- Customize cache allocation policy in order to reduce cache interference
- High priority tasks are more critical and thus requires priority in using resources such as CPU and cache.

Main design ideas
- Partition a cache at the granularity of columns
  - No need to change tag size
- Use task priority to allocate cache partitions.
  - Partitions owned by low priority tasks can be used by high priority tasks.
  - Partitions owned by high priority tasks cannot be used by low priority tasks.
  - Partitions are released when tasks are completed.
- Shared columns
  - Prevent that some tasks do not have cache to use.
  - Shared columns can be used by all tasks.

Contribution 5: A novel “prioritized cache” design is presented to reduce CRPD.
Example

- Two applications
  - OFDM
  - A Mobile Robot Control Application (MR)
  - MR has a higher priority than OFDM.

- A 4-way set-associative cache

A prioritized cache with 4 columns

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OFDM runs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MR runs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OFDM is completed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MR is completed.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hardware Design

- A variant of a set associative cache
  - Additional registers
    - Column Priority Table (CPT)
    - Column Owner Table (COT)
    - Column Sharing Register (CSR)
    - Current Task Register (CTR)
    - Current Task Priority Register (CTPR)
  - Customized Controller
Software Support

- APIs for direct control
  - Set_tid_pri();
  - Set_column_shared()
  - Release_column();

- Modify the OS slightly for transparent support

```c
void reschedule(void)
{
    disable_interrupt();
    ...
    contextswitch();
    Set_tid_pri(current_task->tid,current_task->pri);
    ...
    enable_interrupt();
}
```
WCRT Analysis for a Prioritized Cache

- WCRT estimate formula remains the same.

\[
R_i^0 = C_i; \\
R_i^1 = C_i + \sum_{j=0}^{i-1} \left[ \frac{R_i^0}{P_j} \right] \times (C_j + CRPD(T_i, T_j) + 2C_{cs}) \\
\vdots \\
R_i^k = C_i + \sum_{j=0}^{i-1} \left[ \frac{R_i^{k-1}}{P_j} \right] \times (C_j + CRPD(T_i, T_j) + 2C_{cs})
\]

- \( C_{pre}(T_i, T_j) \) (CRPD) depends on cache partitions.
  - Not using shared cache partitions: CRPD=0
  - Using shared cache partitions: CRPD analysis.
  - Include CRPD in WCRT

- Comparison with benchmarking
  - The worst case
  - Safer for real-time systems

© 2005 Georgia Institute of Technology
OUTLINE

- Motivation
- Problem Statement
- Previous Work
- WCRT Analysis
- Prioritized Cache Design
- Experiments for WCRT Analysis
- Experiments for Prioritized Cache
- Conclusions
- Publications
Experiment

- Simulation Architecture
  - ARM9TDMI
  - 32KB 4-way set associative cache, 16 bytes in each cache line
  - Two-level memory hierarchy
  - Atalanta RTOS developed at Georgia Tech
  - Seamless CVE for simulation

![Diagram of the simulation architecture](image-url)
Experiment

Five approaches

- App1 (Busquests-Mataix’s method): All cache lines used by preempting task are reloaded for a preemption.
- App2: Inter-task cache eviction analysis only.
- App3: Intra-task cache eviction analysis only.
- App4: Lee’s Approach
- App5: Intra-task cache eviction analysis, Inter-task cache eviction analysis plus path analysis (our approach).
Experiment I

- A mobile robot application with three tasks (GTMRL)
  - Edge Detection (ED)
  - Mobile Robot control (MR)
  - OFDM for communication

<table>
<thead>
<tr>
<th>Task</th>
<th>Period (us)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFDM</td>
<td>40000</td>
<td>4</td>
</tr>
<tr>
<td>ED</td>
<td>6500</td>
<td>3</td>
</tr>
<tr>
<td>MR</td>
<td>3500</td>
<td>2</td>
</tr>
</tbody>
</table>
Results of Experiment I

- Three types of preemption
  - ED preempted by MR
  - OFDM preempted by MR
  - OFDM preempted by ED

- Estimate of the number of cache lines to be reloaded

<table>
<thead>
<tr>
<th>preemptions</th>
<th>App.1</th>
<th>App.2</th>
<th>App.3</th>
<th>App.4</th>
<th>App.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED by MR</td>
<td>245</td>
<td>134</td>
<td>187</td>
<td>118</td>
<td>88</td>
</tr>
<tr>
<td>OFDM by MR</td>
<td>254</td>
<td>172</td>
<td>187</td>
<td>135</td>
<td>98</td>
</tr>
<tr>
<td>OFDM by ED</td>
<td>245</td>
<td>87</td>
<td>106</td>
<td>85</td>
<td>81</td>
</tr>
</tbody>
</table>
## Results of Experiment I

### WCRT estimates

- A5 vs. A4, up to 24% reduction in WCRT estimate

<table>
<thead>
<tr>
<th>Cache miss penalty</th>
<th>Task</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>OFDM</td>
<td>9847</td>
<td>9771</td>
<td>9789</td>
<td>9764</td>
<td>9684</td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>2567</td>
<td>2409</td>
<td>2428</td>
<td>2407</td>
<td>2403</td>
</tr>
<tr>
<td>20</td>
<td>OFDM</td>
<td>12510</td>
<td>12242</td>
<td>12378</td>
<td>10424</td>
<td>10264</td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>2812</td>
<td>2496</td>
<td>2534</td>
<td>2492</td>
<td>2484</td>
</tr>
<tr>
<td>30</td>
<td>OFDM</td>
<td>23501</td>
<td>19249</td>
<td>17244</td>
<td>12468</td>
<td>12258</td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>3057</td>
<td>2583</td>
<td>2640</td>
<td>2577</td>
<td>2565</td>
</tr>
<tr>
<td>40</td>
<td>OFDM</td>
<td>45216</td>
<td>31284</td>
<td>30532</td>
<td>16952</td>
<td>12966</td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>3302</td>
<td>2670</td>
<td>2746</td>
<td>2662</td>
<td>2646</td>
</tr>
</tbody>
</table>
Results of Experiment I

Cache miss penalty changes from 10 clock cycles to 80 clock cycles
WCRT estimate reduction up to 28%
Experiment II

- DSP application
  - Adaptive Differential Pulse Coding Modulation Coder (ADPCMC)
  - ADPCM Decoder (ADPCMD)
  - Inverse Discrete Cosine Transform (IDCT)

<table>
<thead>
<tr>
<th>Task</th>
<th>Period (us)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDCT</td>
<td>4500</td>
<td>2</td>
</tr>
<tr>
<td>ADPCMD</td>
<td>10000</td>
<td>3</td>
</tr>
<tr>
<td>ADPCMC</td>
<td>50000</td>
<td>4</td>
</tr>
</tbody>
</table>
Results of Experiment II

- Three types of preemption
  - ADPCMD preempted by IDCT
  - ADPCMC preempted IDCT
  - ADPCMC preempted by ADPCMD

- Estimate of the number of cache lines to be reloaded

<table>
<thead>
<tr>
<th>preemptions</th>
<th>App.1</th>
<th>App.2</th>
<th>App.3</th>
<th>App.4</th>
<th>App.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADPCMD by IDCT</td>
<td>249</td>
<td>68</td>
<td>98</td>
<td>64</td>
<td>56</td>
</tr>
<tr>
<td>ADPCMC by IDCT</td>
<td>220</td>
<td>114</td>
<td>98</td>
<td>92</td>
<td>64</td>
</tr>
<tr>
<td>ADPCMC by ADPCMD</td>
<td>183</td>
<td>58</td>
<td>89</td>
<td>55</td>
<td>46</td>
</tr>
</tbody>
</table>
Results of Experiment II

- **WCRT estimates**
  - The number of cache conflicts is small.
  - No big difference between A4 and A5.
  - Cache impact on WCRT relates to the number of cache conflicts and cache miss penalty.

<table>
<thead>
<tr>
<th>Cache miss penalty</th>
<th>Task</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>ADPCMC</td>
<td>35742</td>
<td>35701</td>
<td>35071</td>
<td>35027</td>
<td>34676</td>
</tr>
<tr>
<td></td>
<td>ADPCMD</td>
<td>6565</td>
<td>6315</td>
<td>6377</td>
<td>6309</td>
<td>6291</td>
</tr>
<tr>
<td>20</td>
<td>ADPCMC</td>
<td>48528</td>
<td>38687</td>
<td>37987</td>
<td>35983</td>
<td>34967</td>
</tr>
<tr>
<td></td>
<td>ADPCMD</td>
<td>6931</td>
<td>6431</td>
<td>6555</td>
<td>6419</td>
<td>6383</td>
</tr>
<tr>
<td>30</td>
<td>ADPCMC</td>
<td>88606</td>
<td>39555</td>
<td>39055</td>
<td>38911</td>
<td>38779</td>
</tr>
<tr>
<td></td>
<td>ADPCMD</td>
<td>7297</td>
<td>6547</td>
<td>6733</td>
<td>6529</td>
<td>6475</td>
</tr>
<tr>
<td>40</td>
<td>ADPCMC</td>
<td>359239</td>
<td>48714</td>
<td>47722</td>
<td>39931</td>
<td>39755</td>
</tr>
<tr>
<td></td>
<td>ADPCMD</td>
<td>7663</td>
<td>6663</td>
<td>6911</td>
<td>6639</td>
<td>6567</td>
</tr>
</tbody>
</table>
Results of Experiment II

- Change cache miss penalty from 10 clock cycles to 80 clock cycles
- WCRT estimate reduction up to 18%
- Cache-related WCRT analysis is useful only when cache conflicts have a great impact on WCRT.

![WCRT Estimate of ADPCMC Graph](image)
## Experiment III

### Six Tasks

<table>
<thead>
<tr>
<th>Tasks</th>
<th>MR</th>
<th>IDCT</th>
<th>ED</th>
<th>ADPCMD</th>
<th>OFDM</th>
<th>ADPCMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period (cycles)</td>
<td>7000</td>
<td>9000</td>
<td>13000</td>
<td>20000</td>
<td>40000</td>
<td>50000</td>
</tr>
<tr>
<td>Priorities</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>
### Results of Experiment III

**WCRT Estimate of ADPCMC:** a reduction up to 32% if comparing A5 with A4

<table>
<thead>
<tr>
<th>Cache Miss Penalty</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>51434</td>
<td>34163</td>
<td>34591</td>
<td>33893</td>
<td>33507</td>
</tr>
<tr>
<td>20</td>
<td>75201</td>
<td>51452</td>
<td>57650</td>
<td>38431</td>
<td>34685</td>
</tr>
<tr>
<td>30</td>
<td>232903</td>
<td>59482</td>
<td>74020</td>
<td>58099</td>
<td>38905</td>
</tr>
<tr>
<td>40</td>
<td>--</td>
<td>75073</td>
<td>114209</td>
<td>69495</td>
<td>58142</td>
</tr>
</tbody>
</table>

**WCRT Estimate of OFDM:** a reduction up to 18% if comparing A5 with A4

<table>
<thead>
<tr>
<th>Cache Miss Penalty</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>16901</td>
<td>16551</td>
<td>17050</td>
<td>16496</td>
<td>16330</td>
</tr>
<tr>
<td>20</td>
<td>25904</td>
<td>17199</td>
<td>17242</td>
<td>17001</td>
<td>16757</td>
</tr>
<tr>
<td>30</td>
<td>50831</td>
<td>17847</td>
<td>17750</td>
<td>17699</td>
<td>17184</td>
</tr>
<tr>
<td>40</td>
<td>116464</td>
<td>34694</td>
<td>27718</td>
<td>25615</td>
<td>17611</td>
</tr>
</tbody>
</table>
Results of Experiment III

- Vary cache size from 8KB to 64KB (A4 vs. A5)
- Cache conflict has a bigger impact on WCRT when the cache is small.

![WCRT of ADPCMC](chart1)

![WCRT of OFDM](chart2)
Experiment IV

Show the affects of infeasible preemptions in Lee’s approach

- Use the same tasks specified in Lee’s experiments
- Compute the WCRT with our WCRT estimate formula

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>WCET</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>320,000</td>
<td>$60,234 + 280 \times C_{miss}$</td>
</tr>
<tr>
<td>LUD</td>
<td>1,120,000</td>
<td>$255,998 + 364 \times C_{miss}$</td>
</tr>
<tr>
<td>LMS</td>
<td>1,920,000</td>
<td>$365,893 + 474 \times C_{miss}$</td>
</tr>
<tr>
<td>FIR</td>
<td>25,600,000</td>
<td>$557,589 + 405 \times C_{miss}$</td>
</tr>
</tbody>
</table>

Cache miss penalty = 100 cycles (used in Lee’s experiment)
WCRT of FIR with Lee’s Approach = 5,323,620 cycles
WCRT of FIR with our approach (Approach 5) = 3,778,075 cycles
Reduction in WCRT estimate = 29%
OUTLINE

- Motivation
- Problem Statement
- Previous Work
- WCRT Analysis
- Experiments for WCRT Analysis
- Prioritized Cache Design
- Experiments for Prioritized Cache
- Conclusions
- Publications
Experiment

- Six tasks

<table>
<thead>
<tr>
<th>Tasks</th>
<th>MR</th>
<th>IDCT</th>
<th>ED</th>
<th>ADPCMD</th>
<th>OFDM</th>
<th>ADPCMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period (cycles)</td>
<td>7000</td>
<td>9000</td>
<td>13000</td>
<td>20000</td>
<td>40000</td>
<td>50000</td>
</tr>
<tr>
<td>Priorities</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

- Cache miss penalty: 30 clock cycles

- Cache parameters:
  - 32KB, 16 bytes in each cache line
  - 8 ways, 2 way shared
Experiment

- WCRT estimate of high priority tasks are reduced by up to 26% (our WCRT analysis approach)
- WCRT of low priority tasks increases.

WCRT Estimate with SA and PC

<table>
<thead>
<tr>
<th>Tasks</th>
<th>WCRT (Normalized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR</td>
<td>0.74</td>
</tr>
<tr>
<td>IDCT</td>
<td>0.93</td>
</tr>
<tr>
<td>ED</td>
<td>0.9</td>
</tr>
<tr>
<td>ADPCM</td>
<td>0.89</td>
</tr>
<tr>
<td>OFDM</td>
<td>1.04</td>
</tr>
<tr>
<td>ADPCMC</td>
<td>1.72</td>
</tr>
</tbody>
</table>

SA | PC
OUTLINE

- Motivation
- Problem Statement
- Previous Work
- WCRT Analysis
- Experiments for WCRT Analysis
- Prioritized Cache Design
- Experiments for Prioritized Cache
- Conclusions
- Publications
Five Major Contributions

- A novel approach is proposed to analyze inter-task cache interference.
- Inter-task cache interference analysis is integrated with intra-task cache interference analysis.
- Path analysis is used to improve cache interference analysis.
- A new WCRT estimate formula is proposed.
- A novel “prioritized cache” design is presented to reduce CRPD.
Conclusion

- Our WCRT analysis approach can tighten WCRT estimates effectively.
  - More precise in estimating CRPD
  - No overestimate caused by infeasible preemptions
  - Less complex in computation

- The prioritized cache can reduce cache conflicts significantly.
  - Easy to use
  - Cache behavior simplified
  - WCRT of high priority tasks tightened at the cost of the performance of low priority tasks
Publications Accepted and/or in Print


Publications Under Review

Thank you!
Previous Work: WCRT Analysis

- **ILP based approach [Tomiyama]**
  - Only address a direct mapped instruction cache
  - Data cache and set associative caches are not considered.

- **Comparison with prior work**
  - CRPD is included in WCRT analysis.
  - CRPD is tightened significantly.
    - Inter-task and intra-task cache eviction analysis
    - Path analysis
  - Our approach can be applied to direct mapped cache and set associative cache, or instruction cache and data cache.
Previous Work: Customize Cache Usage (1)

- **Hardware approaches**
  - SMART (Strategic Memory Allocation for Real-Time Systems) Cache [Kirk]
    - Assign cache lines to tasks according to their CPU utilization
  - Column Cache [Suh and Rudolph]
    - Cache is partitioned at the granularity of cache columns
    - Data cache only
  - Lock Cache [Maki]
    - Specific instructions are used to lock each individual data
    - Not easy to use in instruction caches
  - Data Cache Lock [Vera]
  - Split Cache [Juan]
    - Partition a cache at a very fine granularity (as small as one cache line)
    - More hardware overhead (increased tag size etc.)

- **Compare the prioritized cache with prior hardware approaches**
  - Partition a cache at the granularity of columns – No need to change tag size
  - Assign cache partitions according to task priorities
    - High priority tasks are more critical.
  - Easy usage
    - Minor modification in the OS for transparent support
    - No specific instructions needed
  - Apply to instruction caches and data caches
  - Formal WCRT analysis
Previous Work: Customize Cache Usage

Software Approaches

- Software-based cache partitioning approach [Wolfe]
  - Scatter memory locations used by a task in the address space to avoid cache interference
  - Memory fragmentation issue
  - Compiler support needed [Muller]
  - Additional instruction for memory manipulation generated by the compiler

- Customize memory-to-cache mapping [Wager]
  - Additional instructions introduced to remap memory

- OS-Controlled Cache Predictability [Liedtke]
  - Memory remapping transparently supported by OS

- Combination of hardware cache partitioning and a custom compiler [May]

Compare the prioritized cache with prior software approaches

- Do not need sophisticated modification of OS or compilers
- Do not need to control memory-to-cache mapping directly
  - No problem with pre-compiled libraries
  - No additional memory fragmentation problem
Dynamic Memory Allocation

- **Issues**
  - Memory locations unknown
  - Memory allocation time unpredictable

- **Confining memory locations**
  - Allocate a group of memory regions first with memory address known in advance
  - Confine memory allocation in pre-allocated memory regions
  - Analyze worst case memory-to-cache mapping

- **Using hardware memory management unit**
  - DMMU
  - Memory allocation time known
Benefits of using a prioritized cache

- No need to assume cold cache start except for the first run
  - Warm the cache first
- Reduction of cache interference
  - Reduced CRPD
- WCRT estimate tightened

Cold cache cost (included in WCET)