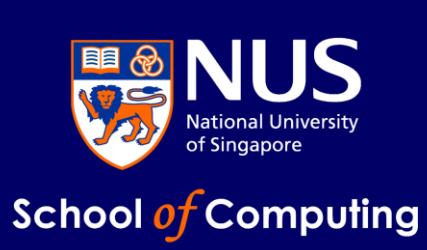
# Shared Cache Aware Task Mapping for WCRT Minimization

**Huping Ding** & Tulika Mitra

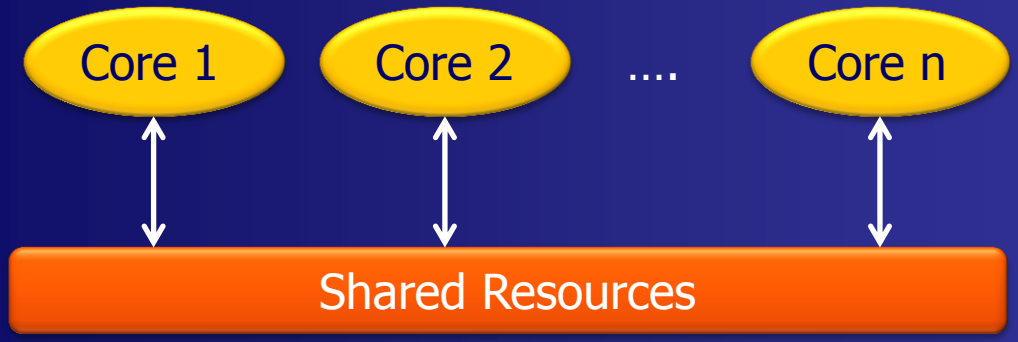
School of Computing,
National University of Singapore

Yun Liang

Center for Energy-efficient Computing and Applications,
School of EECS, Peking University

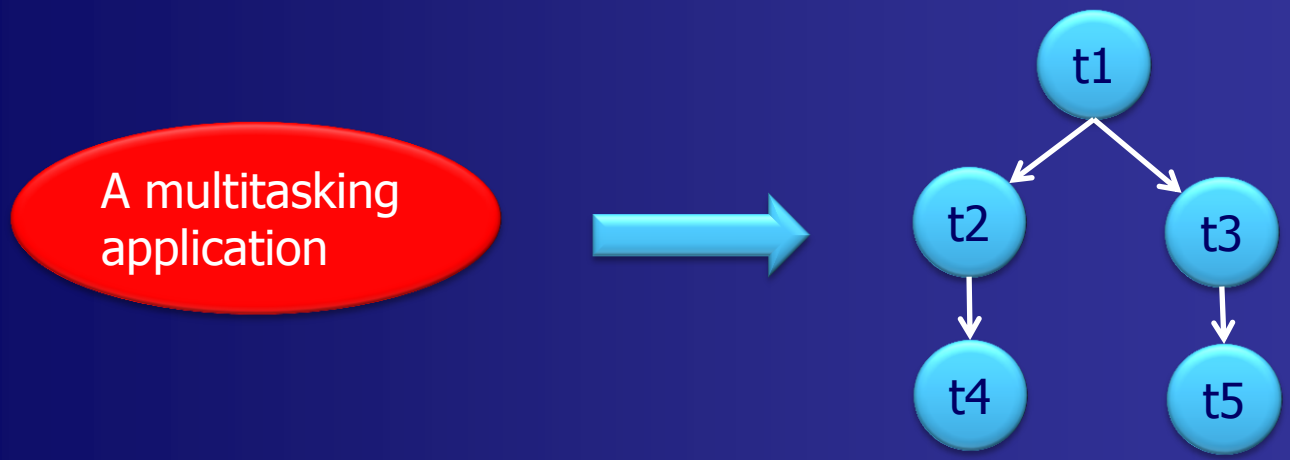



# Multi-core Processor in Embedded Systems




- Embedded systems adopt multi-core processors due to performance, thermal and power constraints
- Embedded systems have real-time constraints, e.g., time deadline of tasks
- Shared Resources, like shared cache, make the timing difficult to estimate, e.g. Worst-Case Response Time (WCRT) analysis

# Worst-Case Response Time



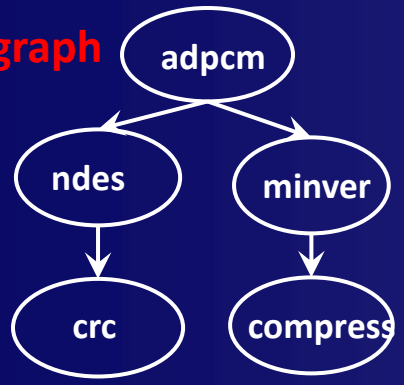

- Tasks can be mapped to any core in a multi-core systems
- The maximal latest finish time among all these tasks is the Worst-Case Response Time (WCRT) of the application
- An important metric for schedulability analysis
- Our purpose is to minimize WCRT in multi-core systems with shared cache via task mapping

# Previous Task Mapping Approaches

- They usually focus on balancing the workload
- They are agnostic to the shared cache behavior
  - State-of-the-art works on shared cache usually fix task mapping before shared L2 cache analysis

# Motivating Example

Task graph

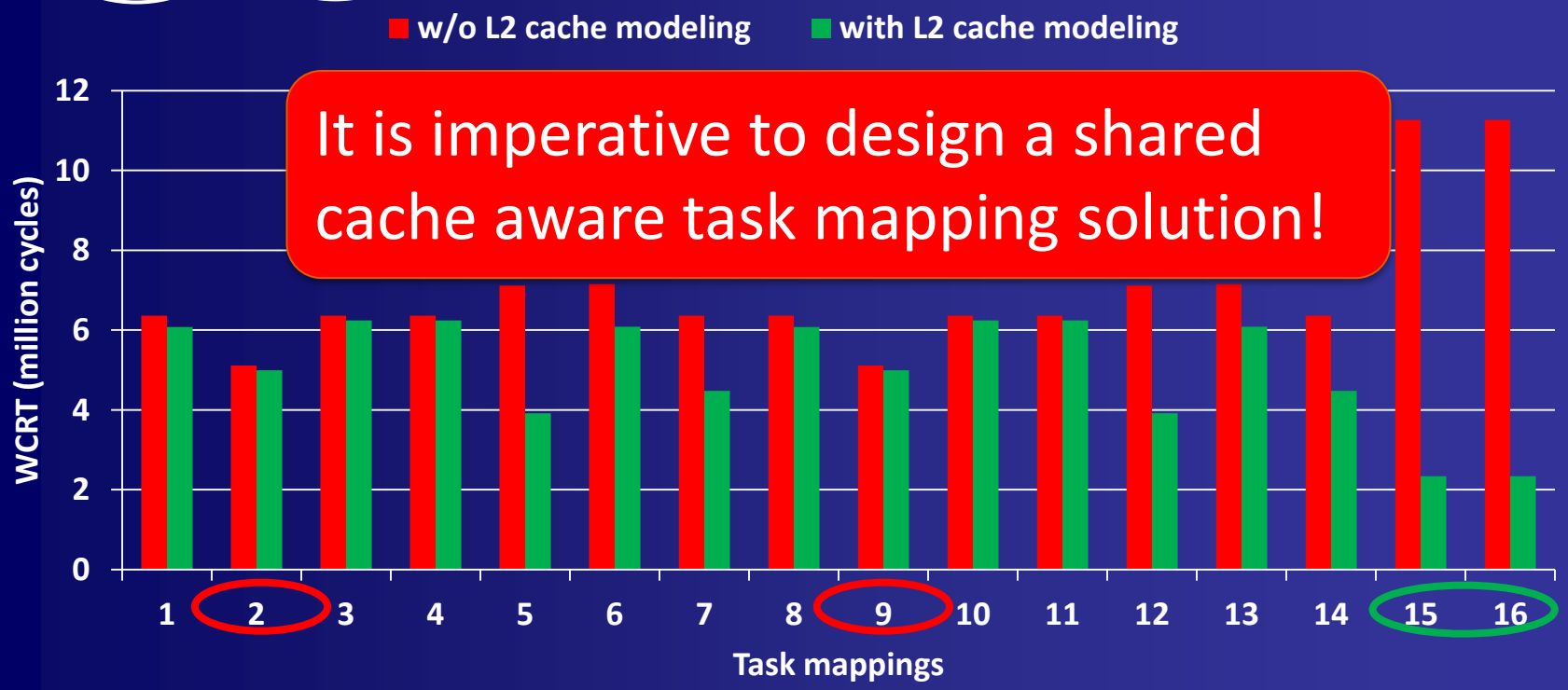



**Configuration:**

2-core, private L1 cache: 256 bytes, shared L2 cache: 2KB

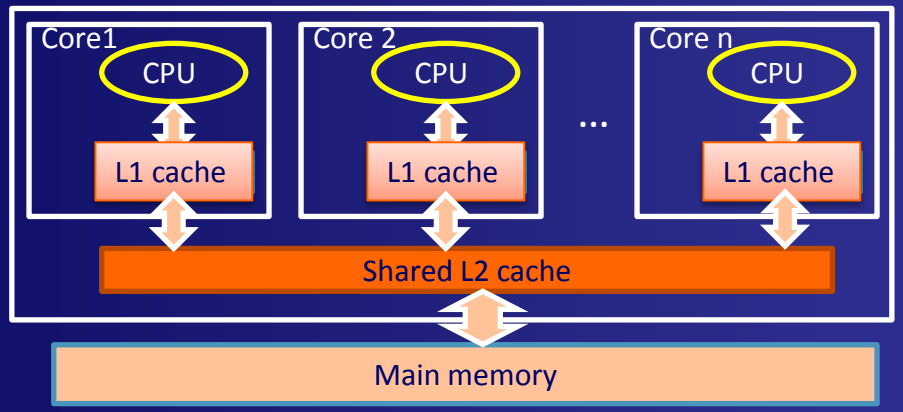
**w/o L2 cache modeling:**

Assume cache miss (hit) for each L2 access in the worst (best) case



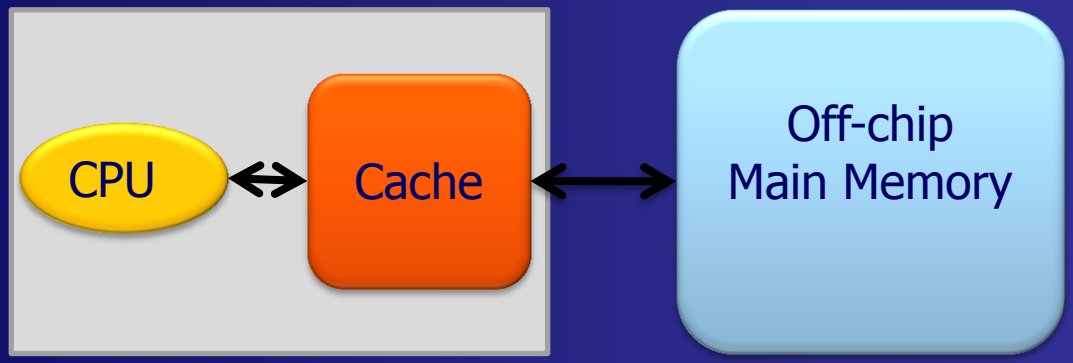

It is imperative to design a shared cache aware task mapping solution!

# Our Architecture




- A private L1 cache for each core
- All cores share a L2 cache
- Tasks execute in a non-preemptive fashion

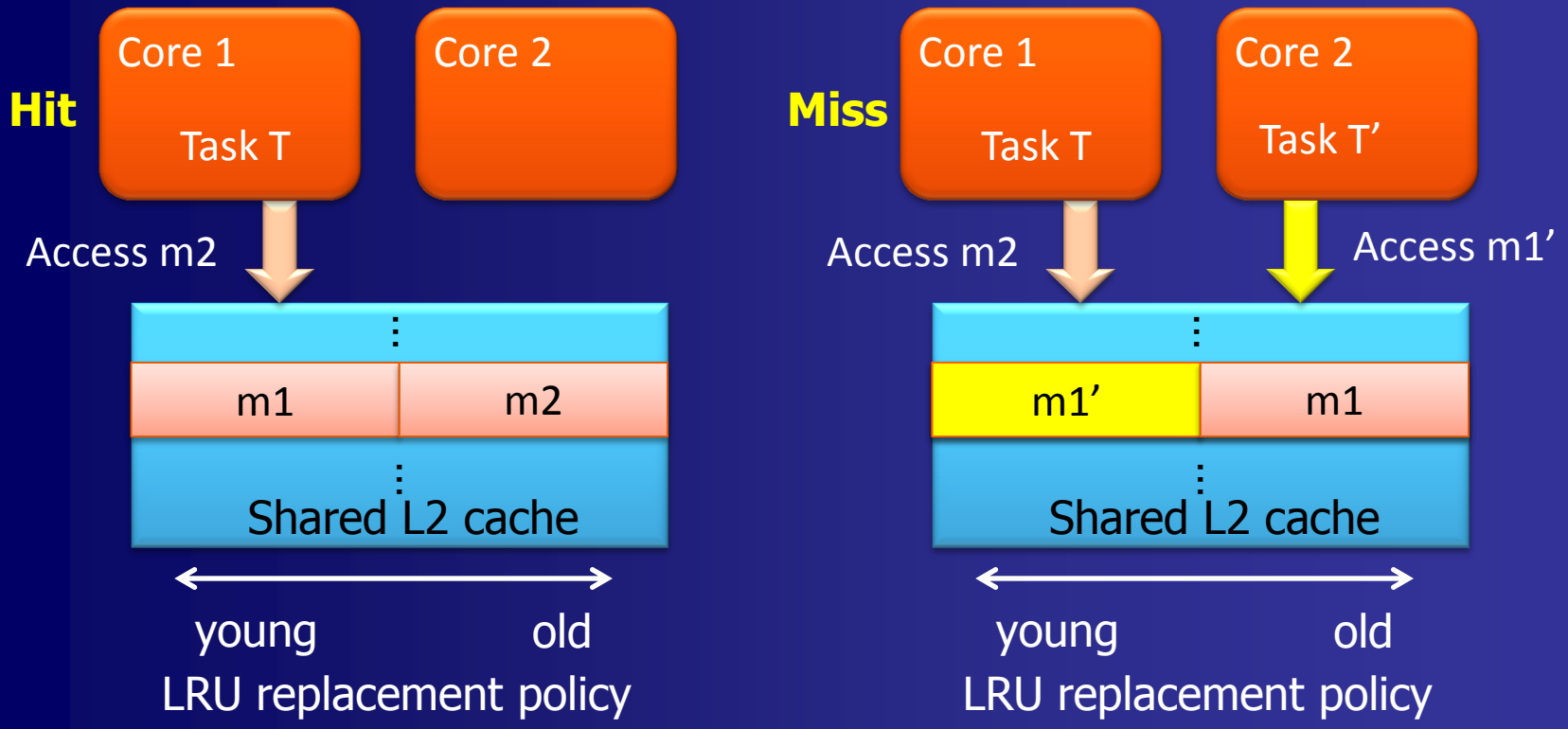
# Private Cache Systems




- Usually only the intra-task analysis is required to estimate the WCET (Worst-Case Execution Time)
- No inter-core cache conflicts

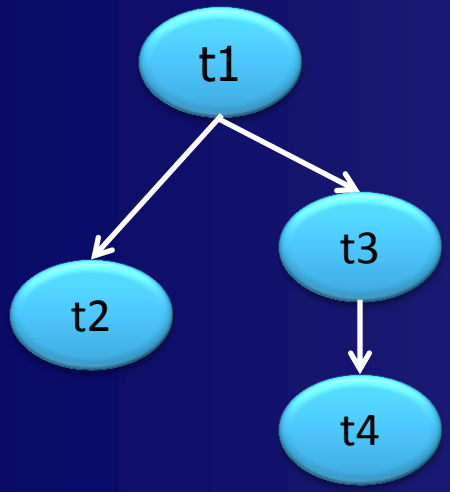
# Shared Cache Multi-core Systems

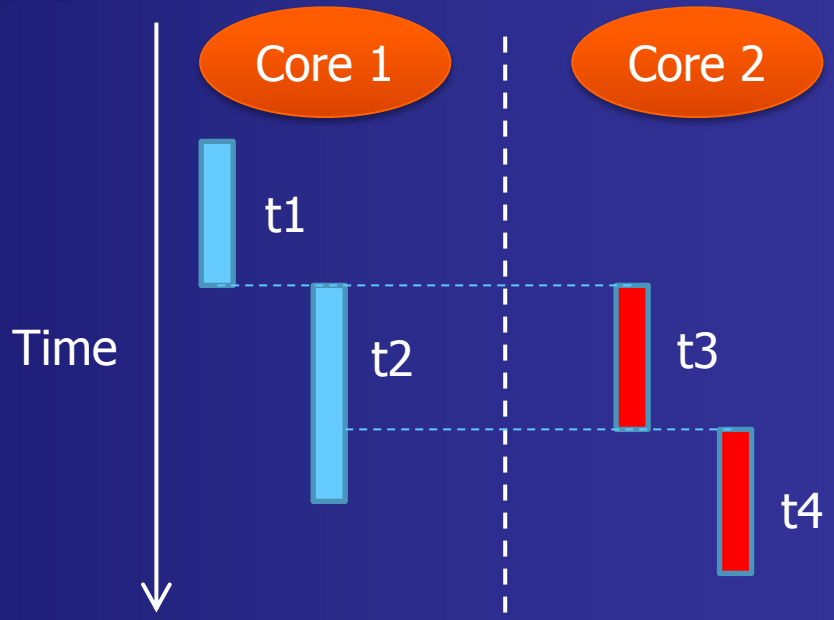
- Inter-core cache conflicts
  - Lead to extra L2 cache misses
- This makes the timing analysis complex



# Task Interference

- Task lifetime : [*Earliest Ready*, *Latest Finish*]
- Interfering tasks : Two tasks mapped to different cores with lifetime overlapped
- Two tasks with dependence between them can never interfere with each other



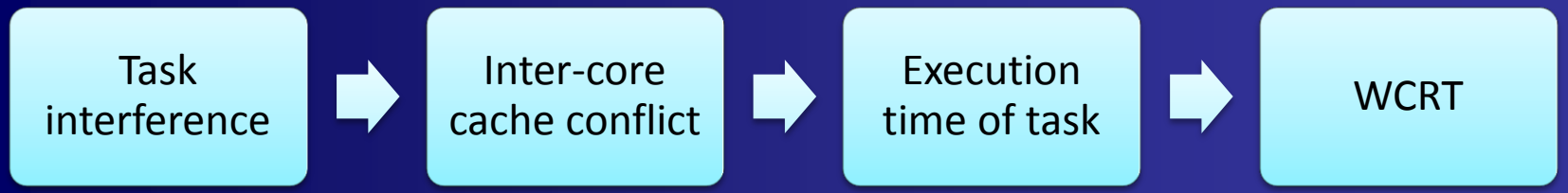

Interfering tasks: t2 and t3, t2 and t4

# Current Efforts on Shared Cache

- They focus on modeling cache and inter-core cache conflicts (*Yan and Zhang RTAS 2008, Hardy and Puaut RTSS 2008 and Liang et al. RTS 2012*)
- Task mapping is fixed (known) before analysis, and agnostic to the shared cache conflicts

# Impact of Task Mapping

- Task interference
  - Two tasks can interfere with each other only when they are mapped to different cores



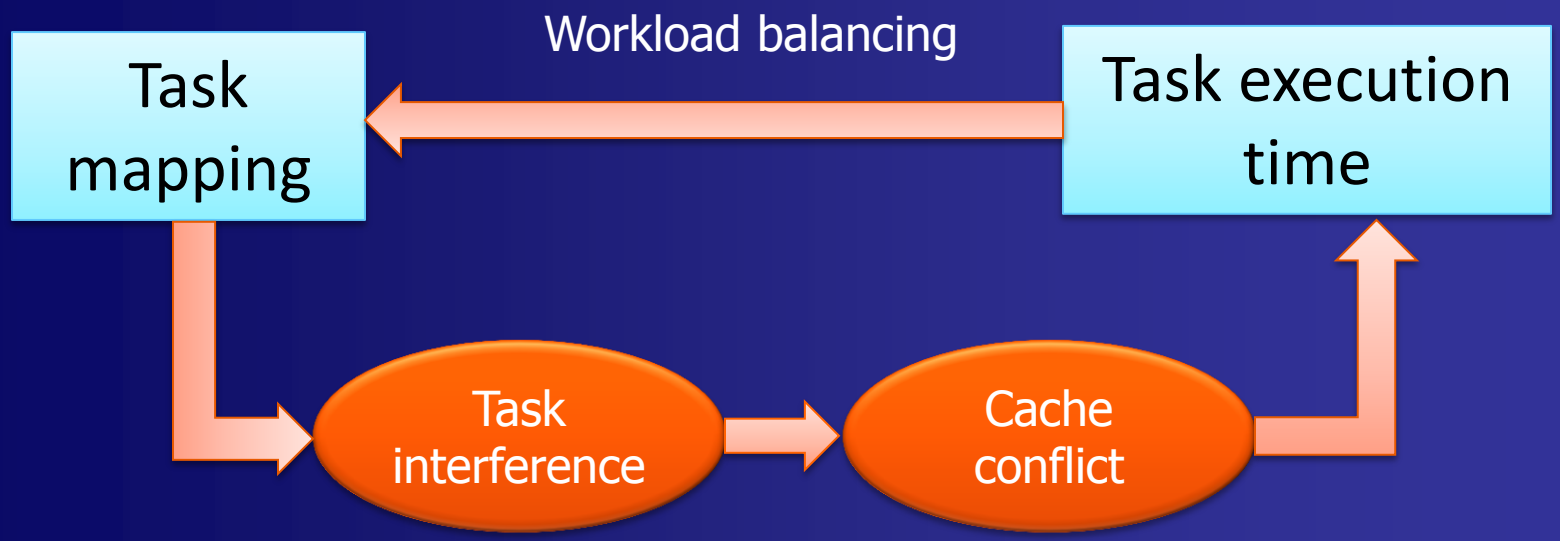

- Workload balance among cores
  - Workload balance influences the total execution time of a multitasking application, thus impacts its WCRT

# Our Aims

- We target the task mapping problem in multi-core systems with shared cache, in order to optimize the WCRT
- We not only consider workload balance, but also minimize inter-core cache conflicts

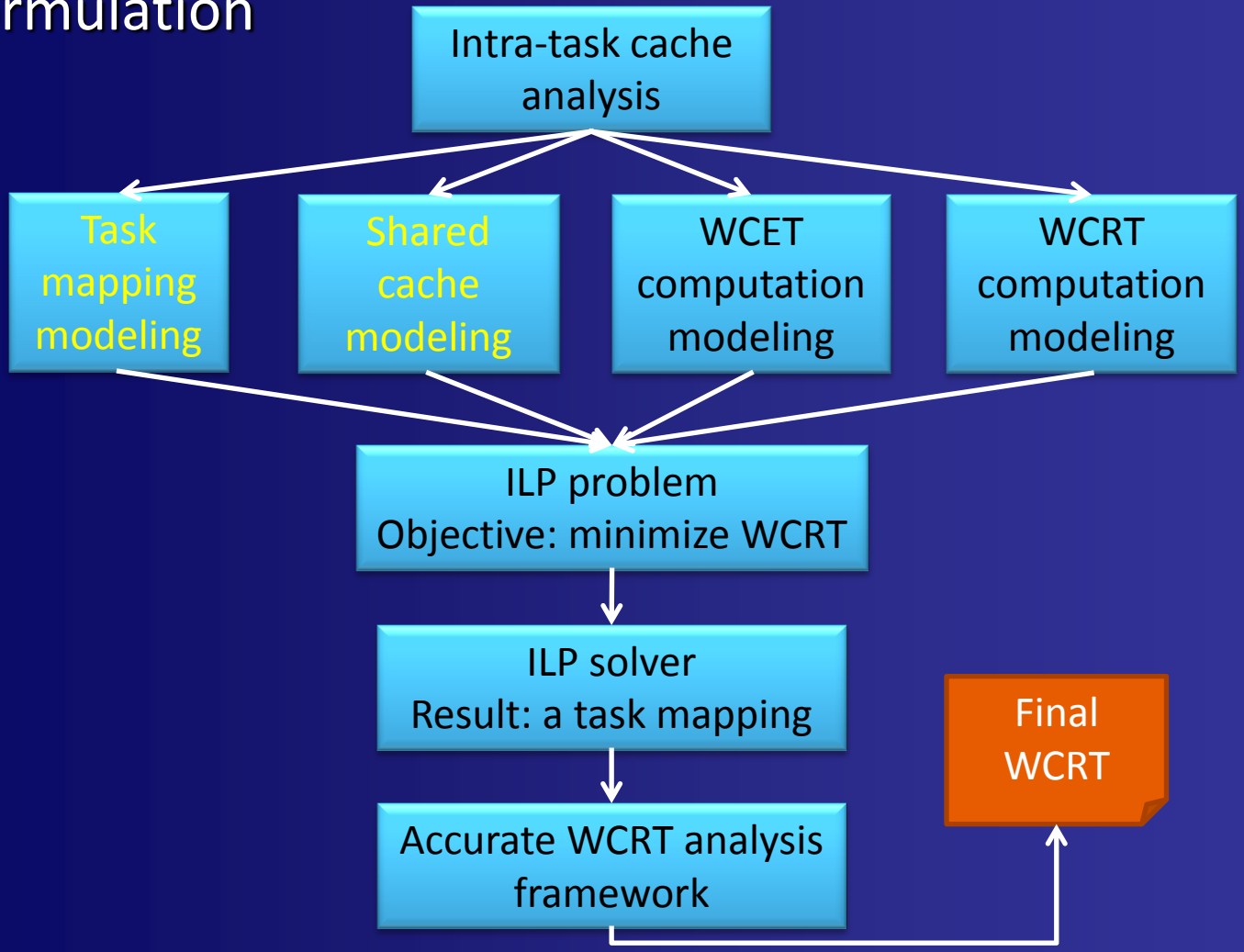
# Challenges

- Exhaustive enumeration of all mappings is impossible as # of cores and # of tasks increase
- Significant complexity due to inter-dependency between task mapping and task execution time



# Our Approach

- An approach based on ILP (integer linear programming) formulation



# Intra-task Analysis

- Must analysis
  - Captures the memory blocks that are guaranteed in the cache
- May analysis
  - Captures the memory blocks that are never in the cache
- Memory block  access classification
  - Always hit
  - Always miss
  - Non-classified

# Task Mapping Modeling

Suppose there are ***N*** tasks and ***M*** cores in the multi-core system. $MAP_{ik}$ is a 0-1 decision variable to indicate if task ***i*** is mapped to core ***k***. Then for any task ***i*** ($0<i\leq N$).

$$\sum_{0<k\leq M} MAP_{ik} = 1$$

For two task ***i*** and ***j***, $SC_{ij}$ is a 0-1 decision variable to indicate if tasks ***i*** and ***j*** are mapped to the same core; $DC_{ij}$ is a 0-1 decision variable to indicate if tasks ***i*** and ***j*** are mapped to different cores

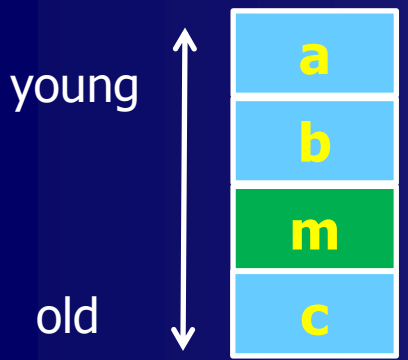
$$SC_{ij} = \begin{cases} 1 & \exists\, 0 < k \leq M, MAP_{ik} = 1 \; and \; MAP_{jk} = 1 \\ 0 & otherwise \end{cases}$$

$$DC_{ij} = 1 - SC_{ij}$$

# Shared Cache Modeling

{***m, a, b, c, d, e***} are mapped to the same set ***s*** in shared L2 cache.
***m, a, b*** and ***c*** are from task ***i***, while ***d*** and ***e*** are tasks ***u*** and ***v*** respectively

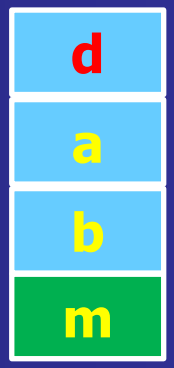
Cache state before considering cache conflicts




LRU replacement policy

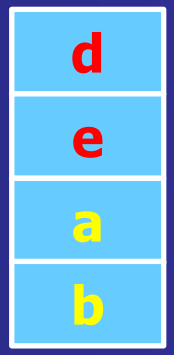
Possible cache states after considering cache conflicts

One cache conflict: **d**
$DC_{iu} = 1$




**Hit**

Two cache conflicts: **d** and **e**
$DC_{iu} = 1$ and $DC_{iv} = 1$




**Miss**

$$(age_m + \sum_{j \in intf(i)} conflict_j^s\ ) \geq A$$

# WCET and WCRT Computation

- WCET
  - Intra-task analysis and shared cache modeling
- WCRT
  - WCRT is the summation of WCET value on the critical path

# Approximations in ILP

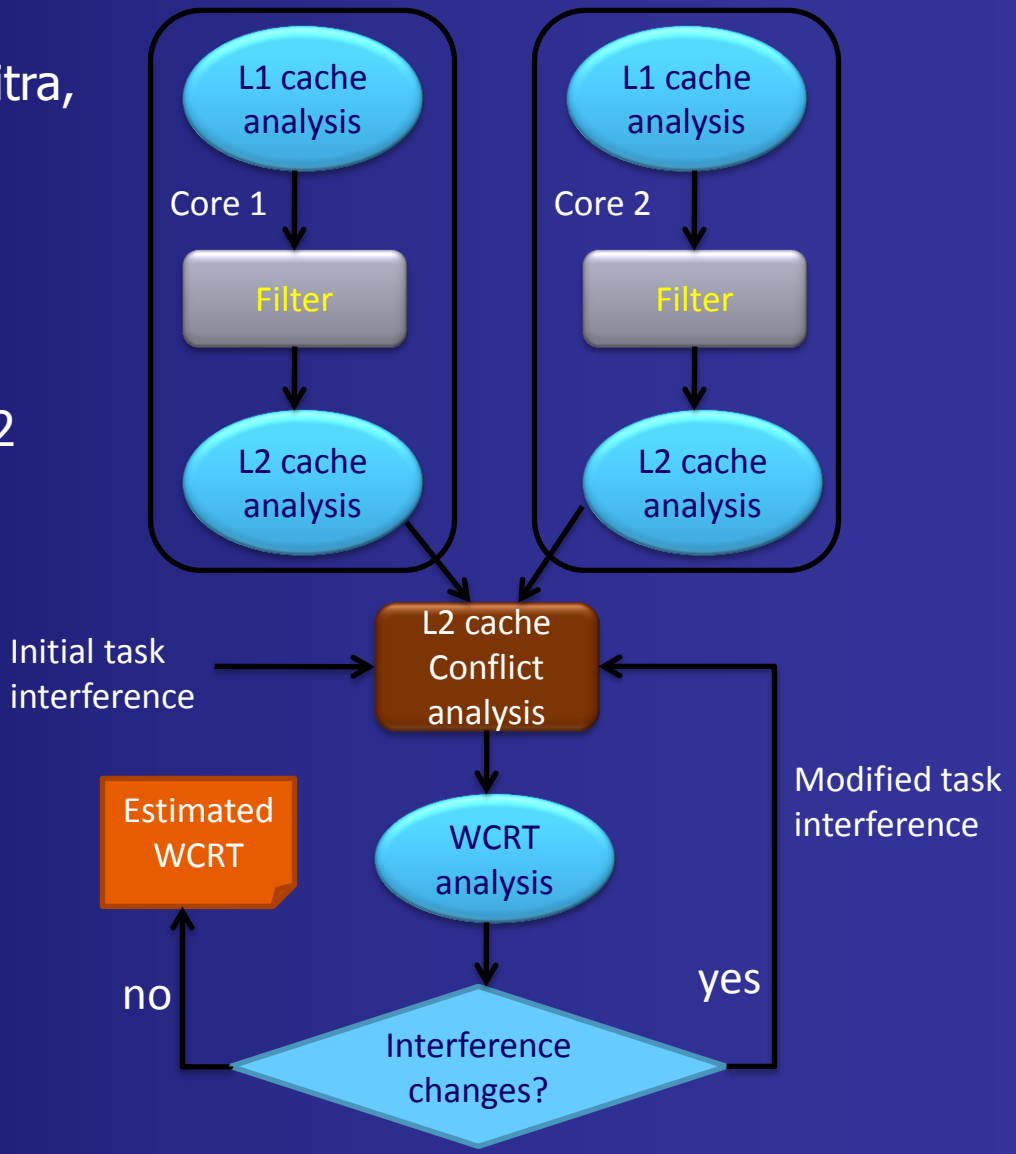
- Interfering tasks
  - Two tasks ***i*** and ***j*** have no dependence and are mapped to different cores

$$interfere_{ij} = DC_{ij}$$

- WCET = Original WCET + Cache conflict penalty
  - Intra-task cache analysis → Original WCET
  - Shared cache modeling → Cache conflict penalty
- Why approximations ?
  - Make the problem easy to solve
- Estimate accurate WCRT with the mapping released by ILP formulation by a WCRT analysis framework

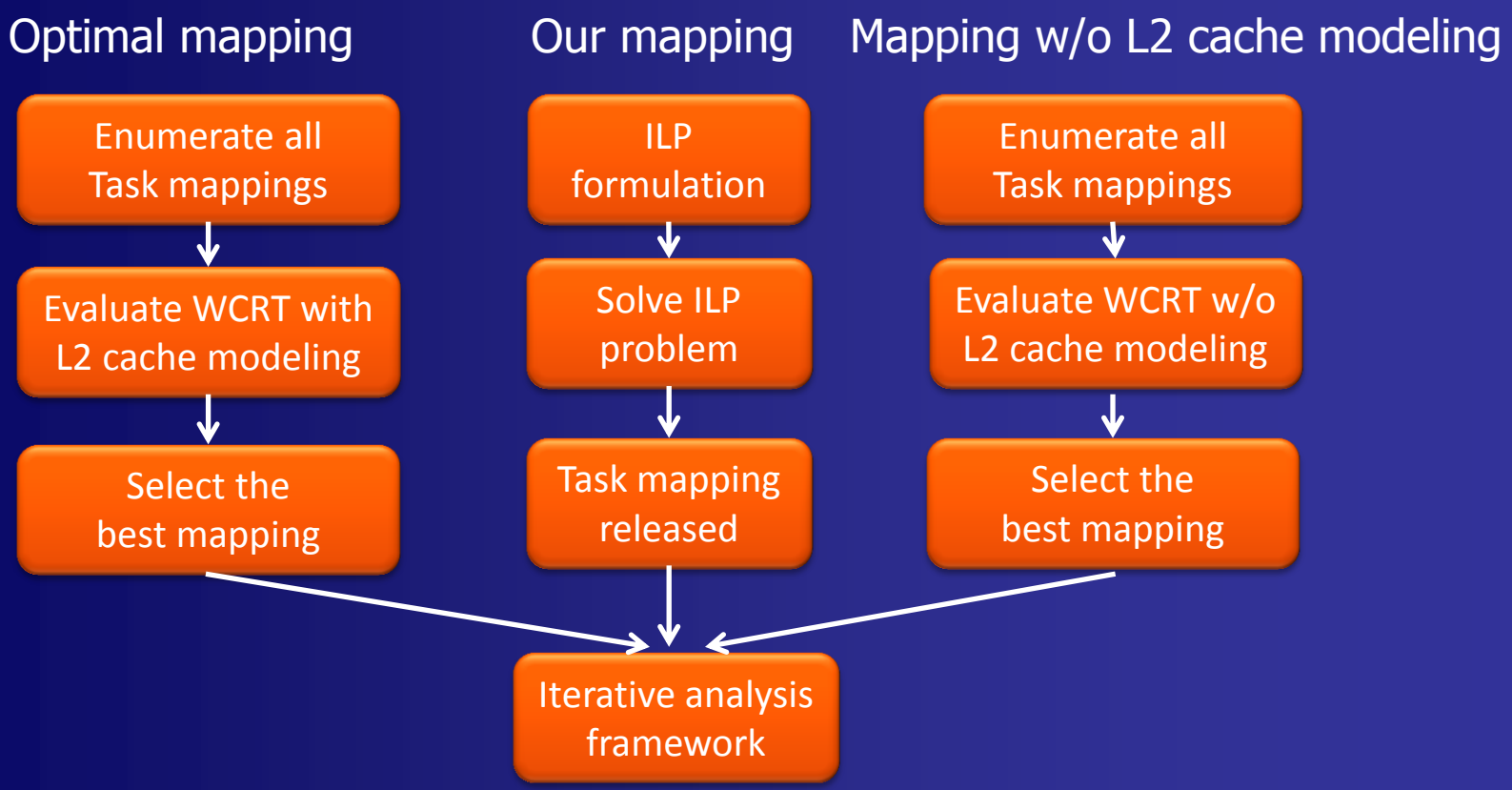
# Iterative Analysis Framework

Yun Liang, Huping Ding, Tulika Mitra, Abhik Roychoudhury, Yan Li, Vivy Suhendra
Timing Analysis of Concurrent Programs Running on Shared Cache Multi-Cores.
In Real-Time System Journal 2012

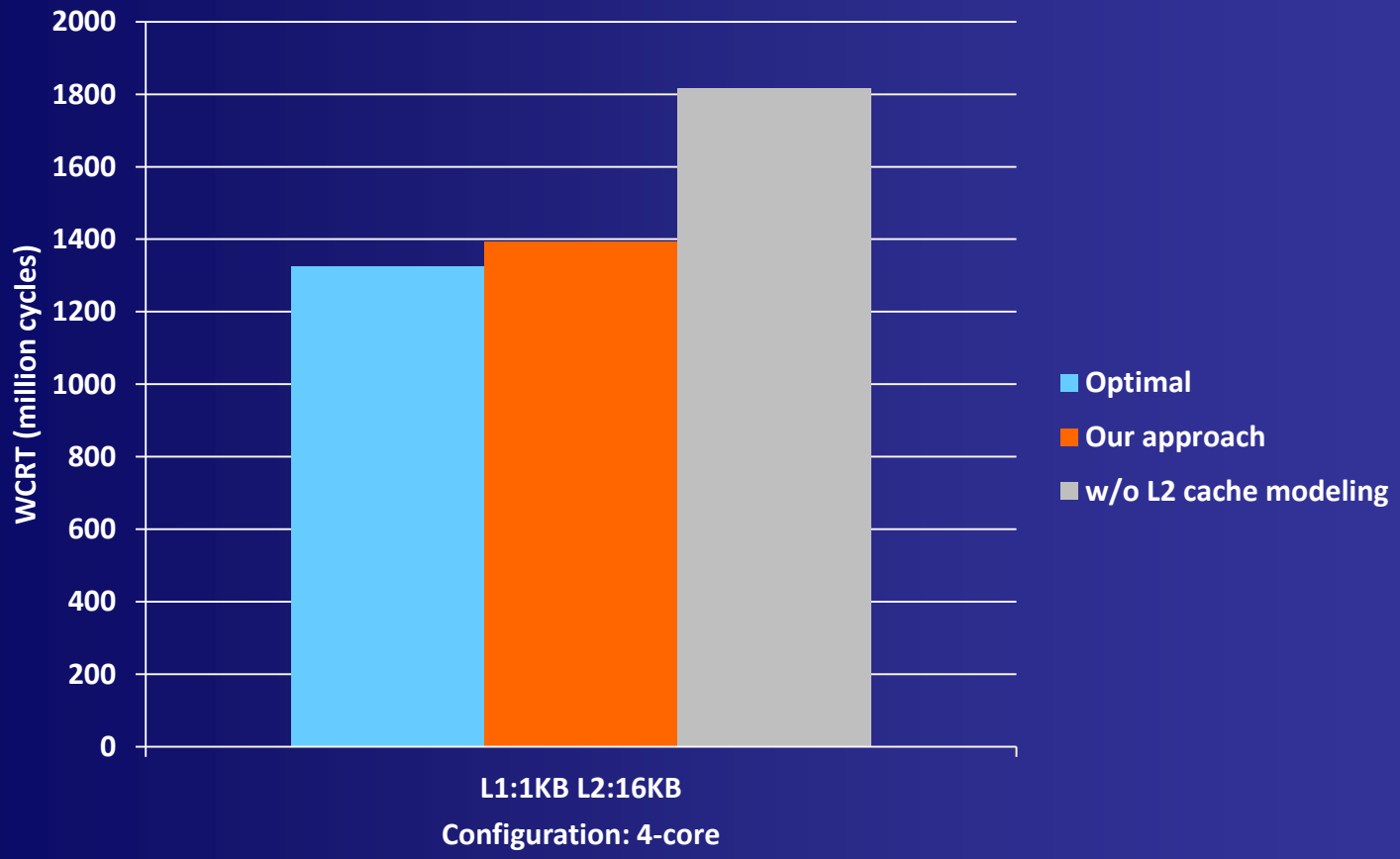


# Experimental Evaluation

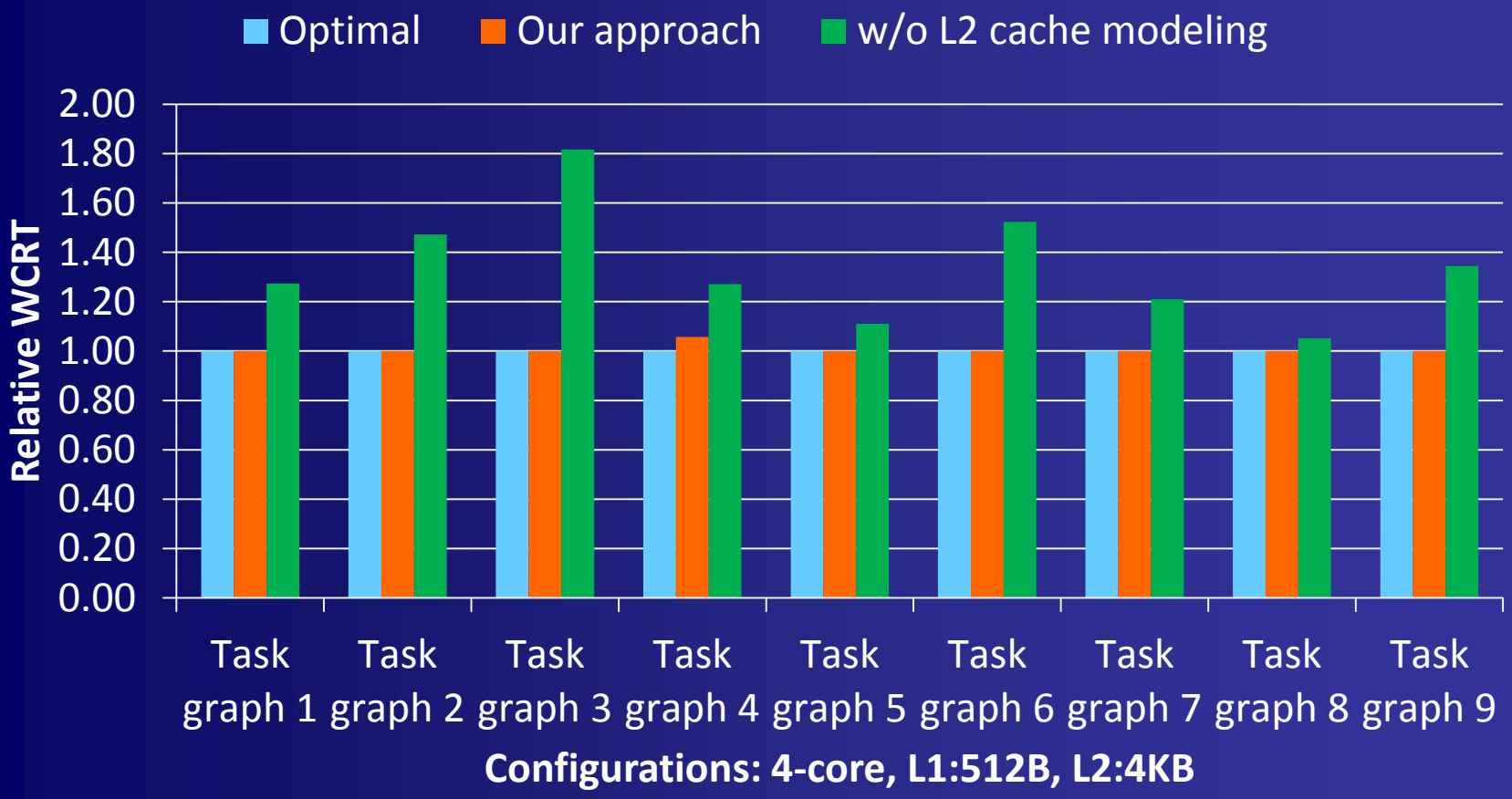
- Real-world benchmark: DEBIE benchmark
- Synthetic benchmarks: MRTC benchmark suite
- Comparison among different approaches



# DEBIE Benchmark Results



# Synthetic Benchmarks Results



# Runtime for Synthetic Benchmarks

| Task graph | # of tasks | Our approach (seconds) | Optimal (seconds) |
|---|---|---|---|
| 1 | 6 | 6.59 | 3.98 |
| 2 | 7 | 33.37 | 1.42 |
| 3 | 8 | 8.77 | 4.93 |
| 4 | 9 | 4.22 | 15.05 |
| 5 | 10 | 17.20 | 67.54 |
| 6 | 11 | 9.27 | 269.29 |
| 7 | 12 | 396.33 | 1089.35 |
| 8 | 13 | 480.62 | 3,883.00 |
| 9 | 14 | 539.31 | 22,794.00 |

# Conclusions

- Tasking mapping has a great impact on WCRT
- An ILP formulation approach that is aware of the shared cache conflict
- Our approach not only considers the workload balance but also minimizes inter-core cache conflict
- Significant reduction in WCRT compared to traditional approach agnostic to shared cache conflicts

# Q & A