Dynamic Command Scheduling for Real-Time Memory Controllers

Yonghui Li[1], Benny Akesson[2] and Kees Goossens[1]

[1]Eindhoven University of Technology,

[2]Czech Technical University in Prague

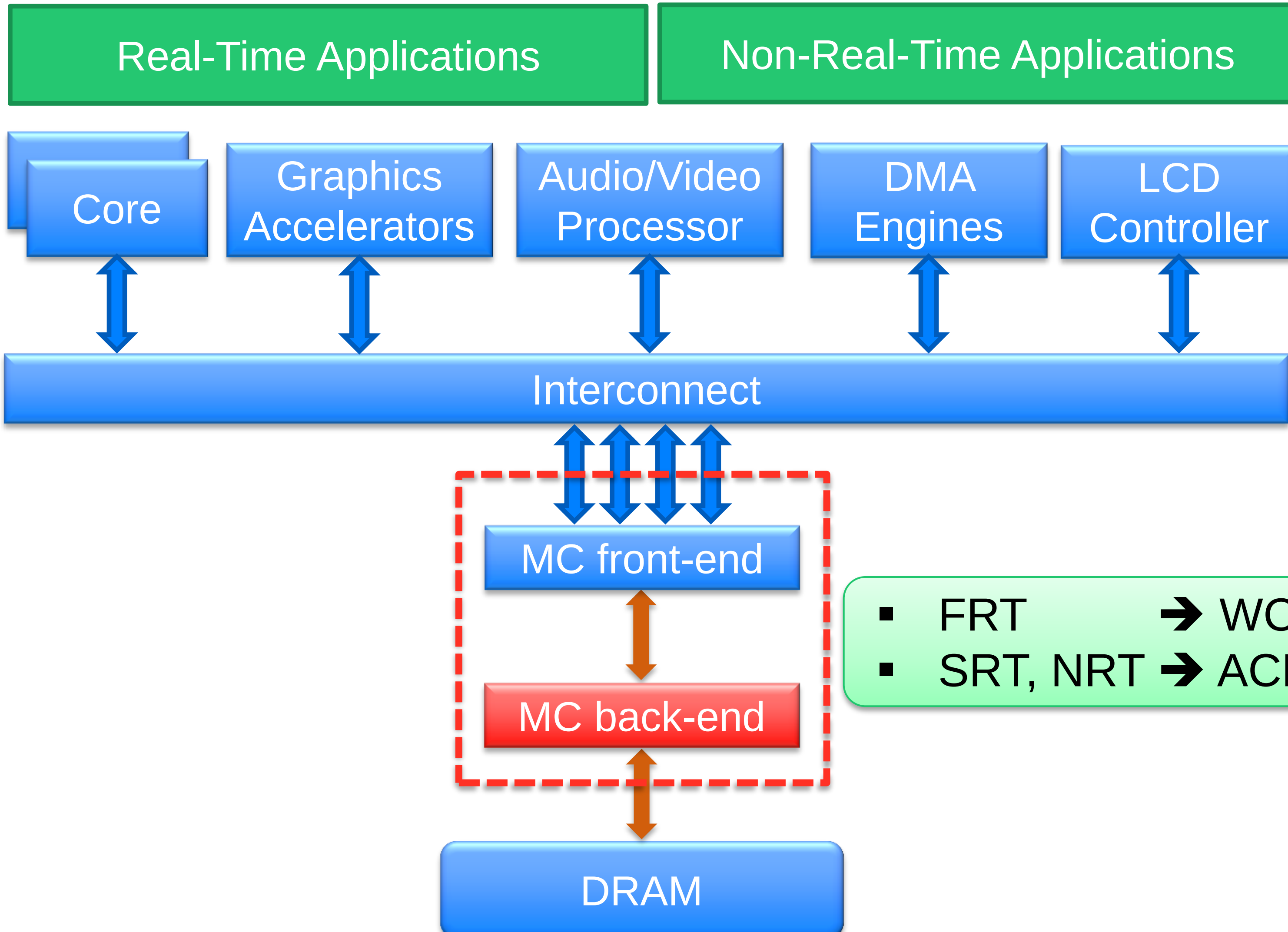
yonghui.li@tue.nl

Czech Technical University in Prague

Technische Universiteit **Eindhoven** University of Technology

Where innovation starts

Mixed Time-Critical Systems



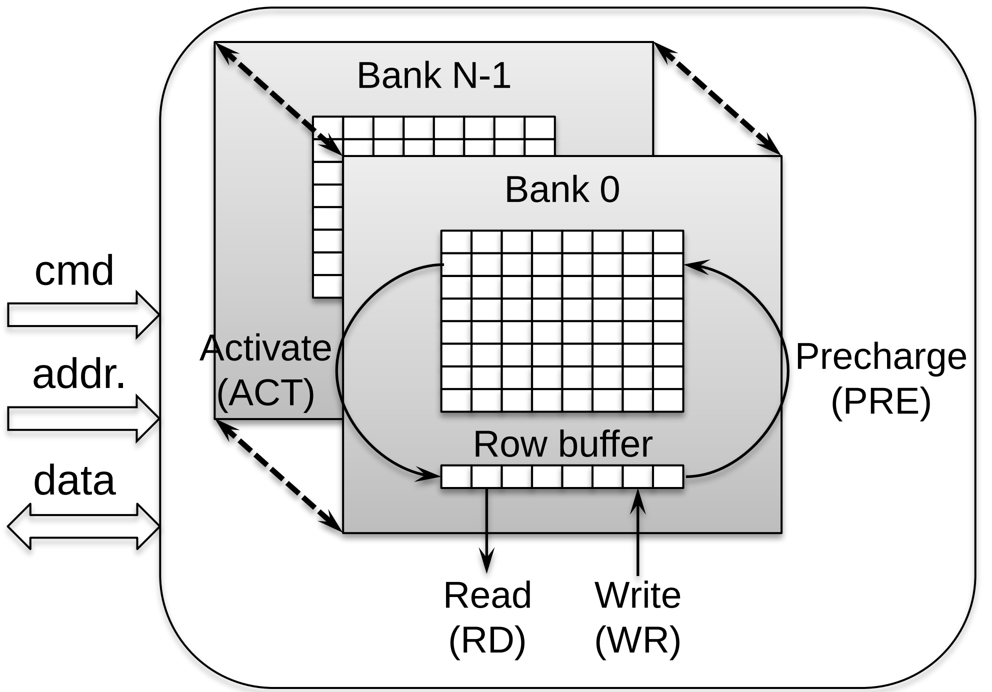
- FRT → WCET
- SRT, NRT → ACET

Outline

- Background
- Architecture and Command Scheduling Algorithm
- Formalization of Dynamic Command Scheduling
- WCET Analysis
- Experiments
- Conclusions

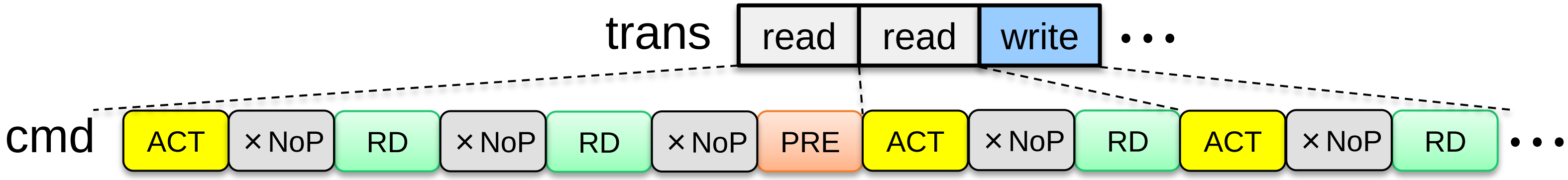
DRAM

- DRAM is accessed by scheduling commands
  - ACT, PRE, RD, WR, REF, NOP
  - subject to timing constraints

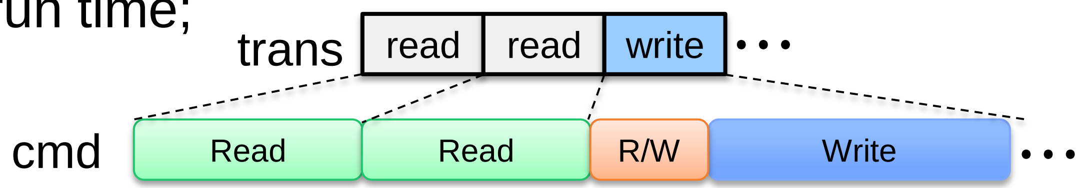


Command Scheduling Approaches

- Static command schedule
  - ➢ analyzable for FRT
  - ➢ not scalable to multiple tasks



- Semi-static command schedule
  - ➢ analyzable and scalable for FRT
  - ➢ limited for a fixed size at run time; worst-case oriented



- Dynamic command schedule
  - ➢ scalable, and good ACET for SRT, NRT
  - ➢ difficult to analyze

Overview

- Goal:
  - guarantee WCET for FRT
  - minimize ACET for SRT, NRT
  - with variable transaction sizes

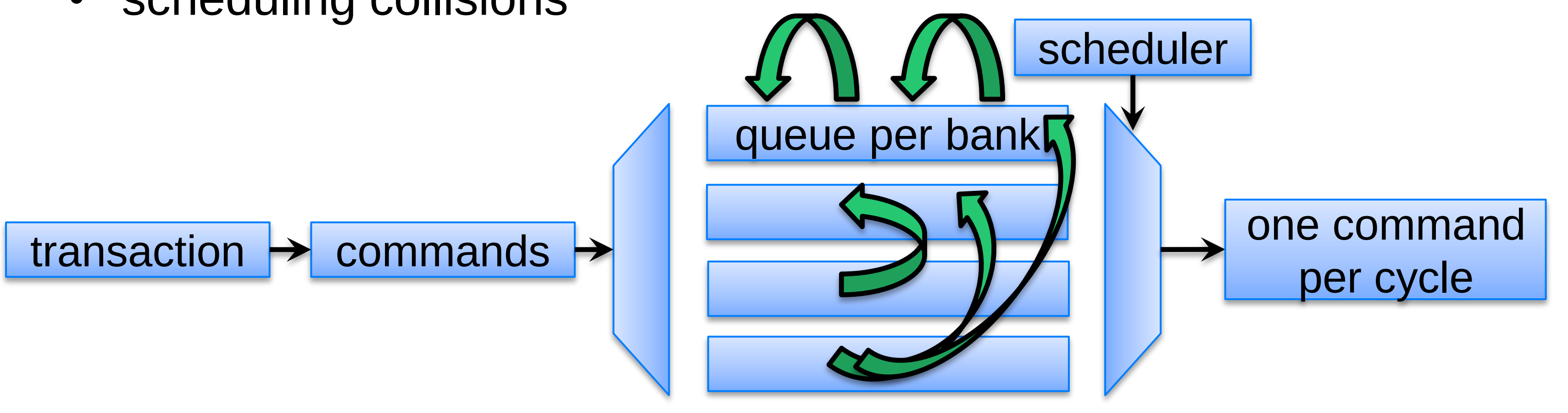
- Contributions
  - to support dynamic command scheduling
  - back-end architecture
  - scheduling algorithm
  - formalization of timing behavior
  - analysis of WCET

Outline

- Background
- Architecture and Command Scheduling Algorithm
- Formalization of Dynamic Command Scheduling
- WCET Analysis
- Experiments
- Conclusions

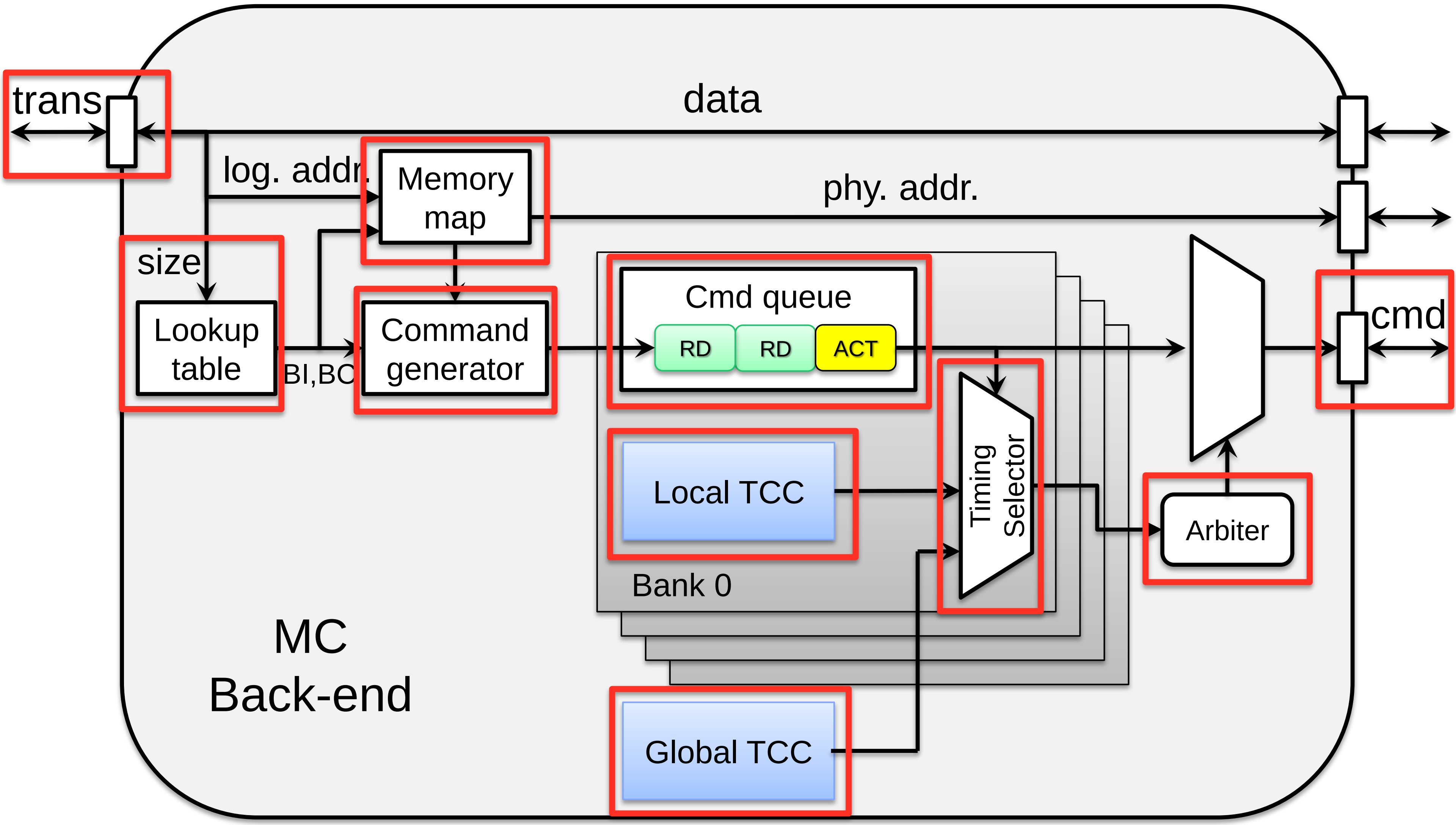
Problem

- Translate a transaction into which sequence of commands
  - different number of commands for variable transaction sizes
    - bank interleaving (BI), burst count (BC) per bank
  - minimum timing constraints between commands impact scheduling order and timing
  - a single scheduler for all commands to any banks
    - scheduling collisions



- Analyzable WCET for variable transaction sizes

Back-End Architecture

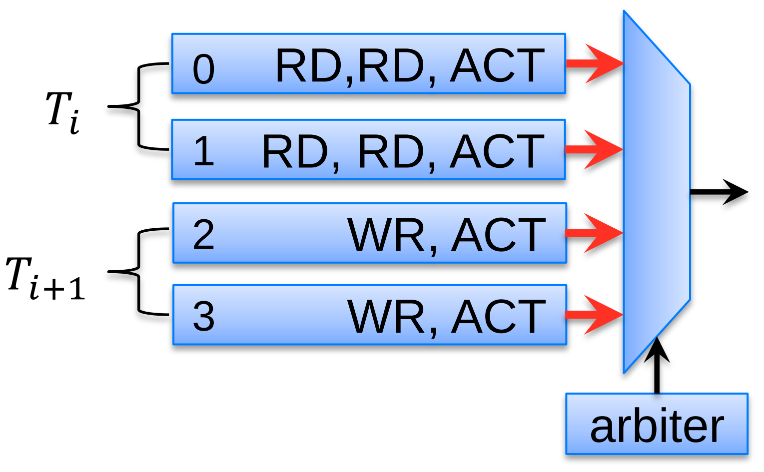


Czech Technical University in Prague

Technische Universiteit Eindhoven University of Technology

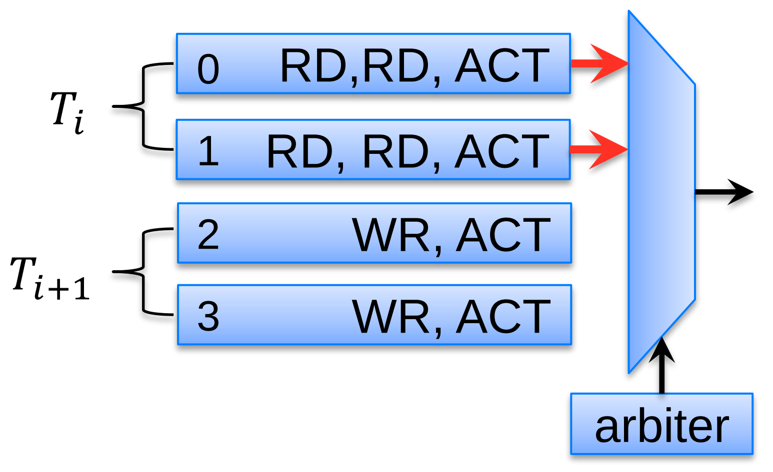
Scheduling Algorithm

- Executes every cycle based on command priorities
- Only used for commands that satisfy their timing constraints



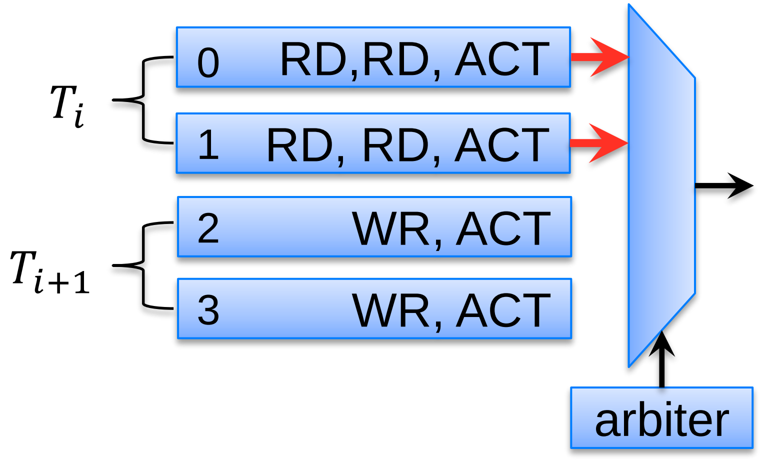
Scheduling Algorithm

- Executes every cycle based on command priorities
- Only used for commands that satisfy their timing constraints
  1. FCFS per transaction



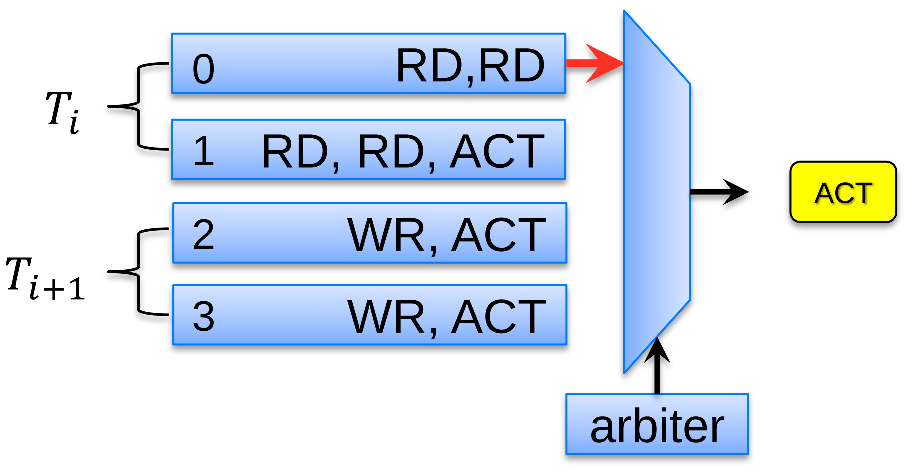
Scheduling Algorithm

- Executes every cycle based on command priorities
- Only used for commands that satisfy their timing constraints
  1. FCFS per transaction
  2. access banks in ascending order per transaction



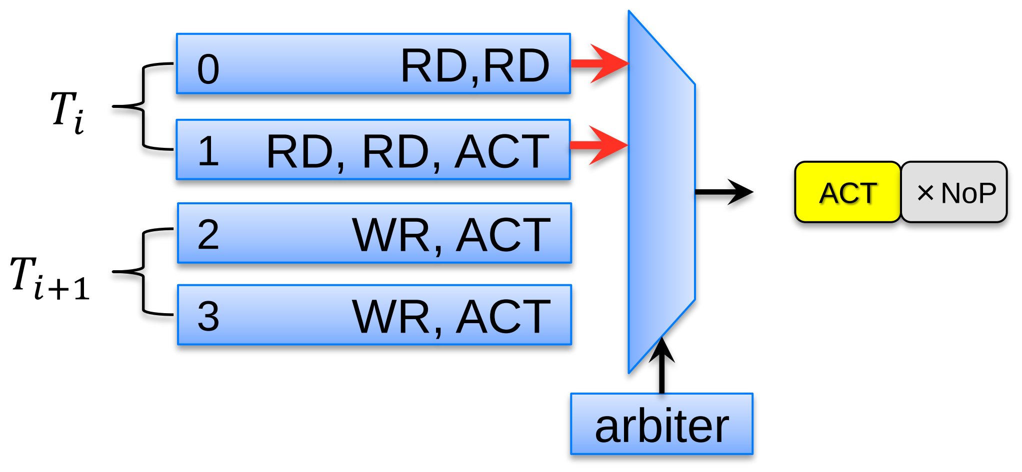
Scheduling Algorithm

- Executes every cycle based on command priorities
- Only used for commands that satisfy their timing constraints
  1. FCFS per transaction
  2. access banks in ascending order per transaction



Scheduling Algorithm

- Executes every cycle based on command priorities
- Only used for commands that satisfy their timing constraints
  1. FCFS per transaction
  2. access banks in ascending order per transaction

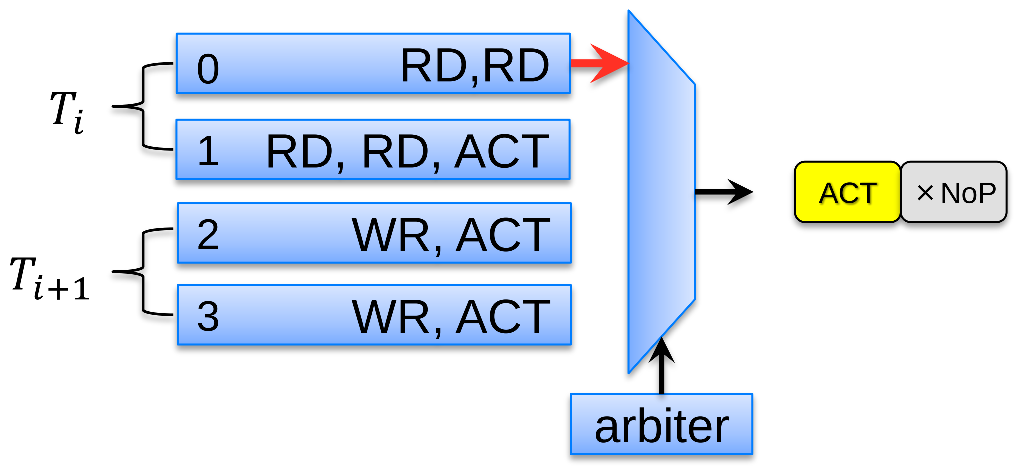


Czech Technical University in Prague

Technische Universiteit Eindhoven University of Technology

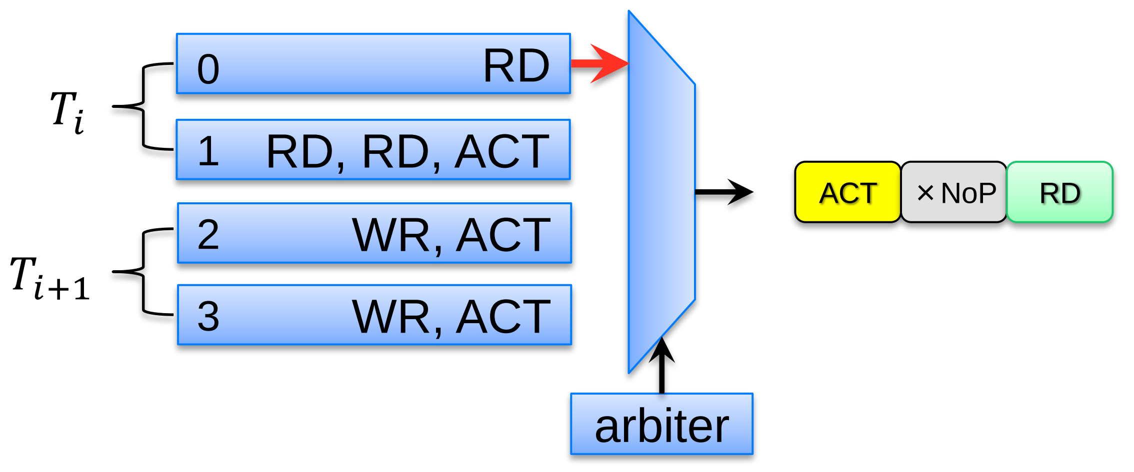
Scheduling Algorithm

- Executes every cycle based on command priorities
- Only used for commands that satisfy their timing constraints
  1. FCFS per transaction
  2. access banks in ascending order per transaction
  3. read/write data before opening another bank



Scheduling Algorithm

- Executes every cycle based on command priorities
- Only used for commands that satisfy their timing constraints
  1. FCFS per transaction
  2. access banks in ascending order per transaction
  3. read/write data before opening another bank

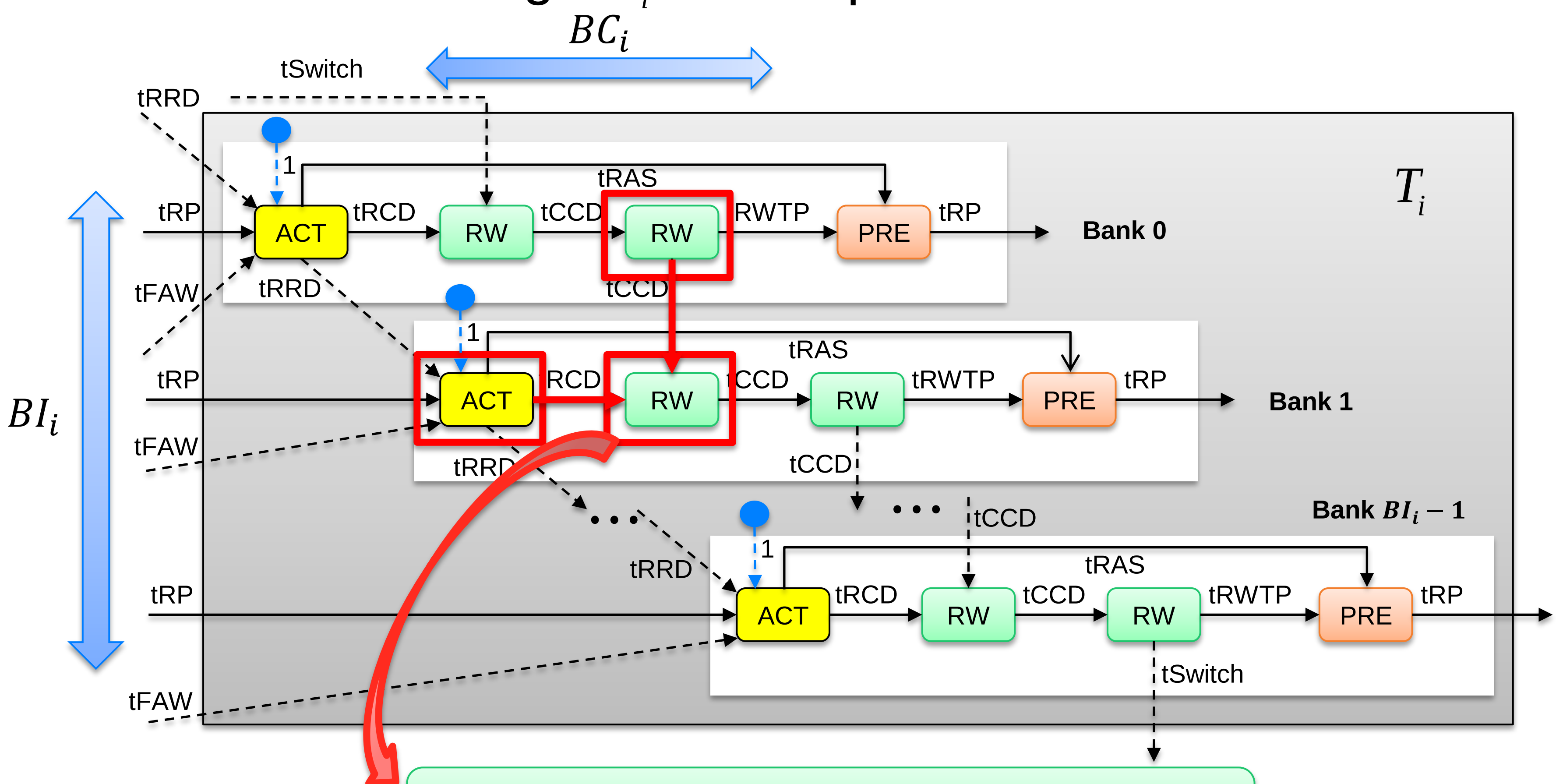


Outline

- Background
- Architecture and Command Scheduling Algorithm
- Formalization of Dynamic Command Scheduling
- WCET Analysis
- Experiments
- Conclusions

Timing Dependencies of a Transaction

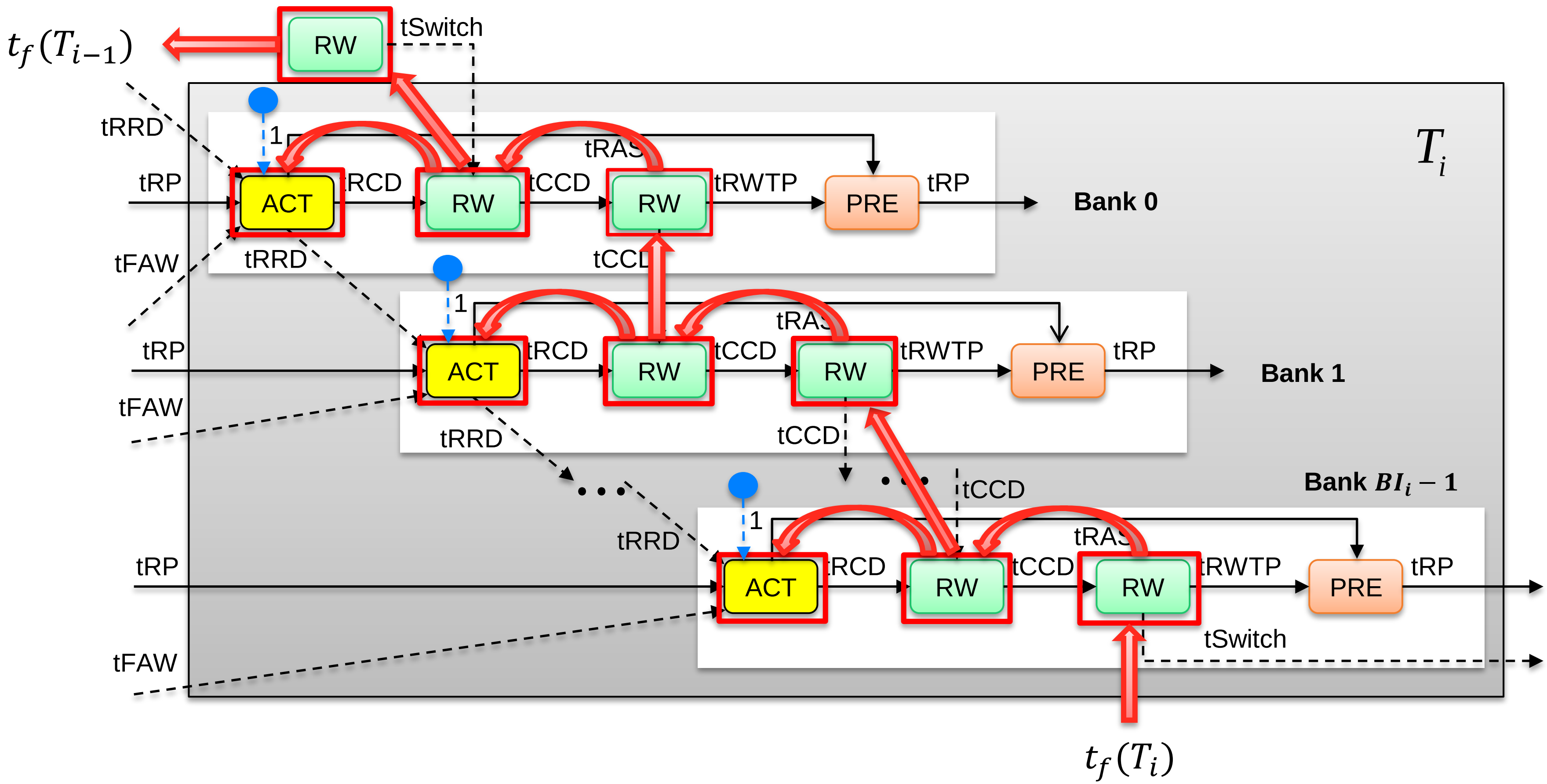
- A transaction T_i is executed by accessing BI_i successive banks and issuing BC_i bursts per bank



$$t(RW) = \max\{t(RW') + tCCD,\ t(ACT) + tRCD\}$$

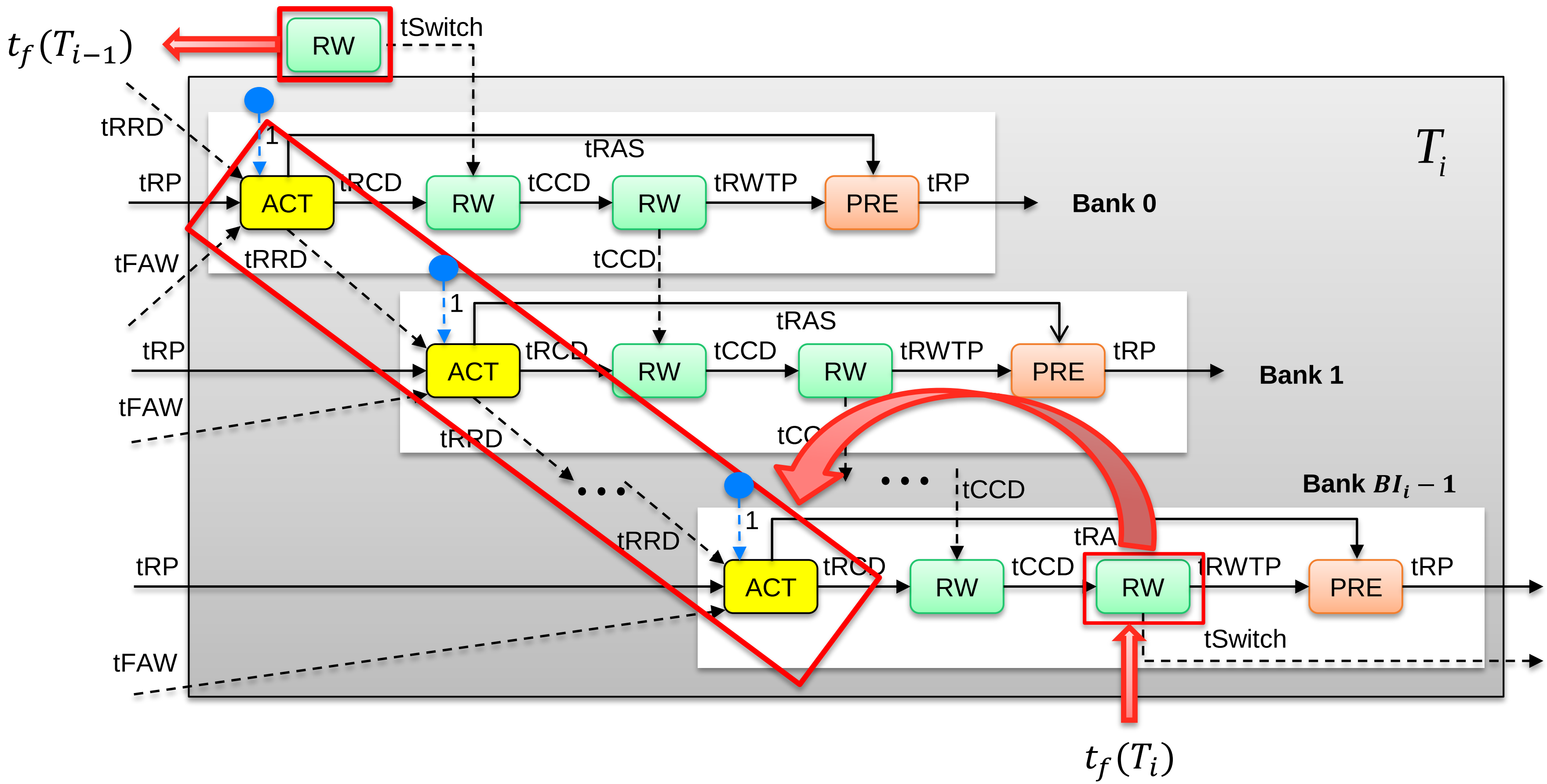
Lemma 1 (Finishing Time)

- The finishing time of T_i depends on the scheduling time of its ACT commands and the finishing time of T_{i-1}



Lemma 1 (Finishing Time)

- The finishing time of T_i depends on the scheduling time of its ACT commands and the finishing time of T_{i-1}

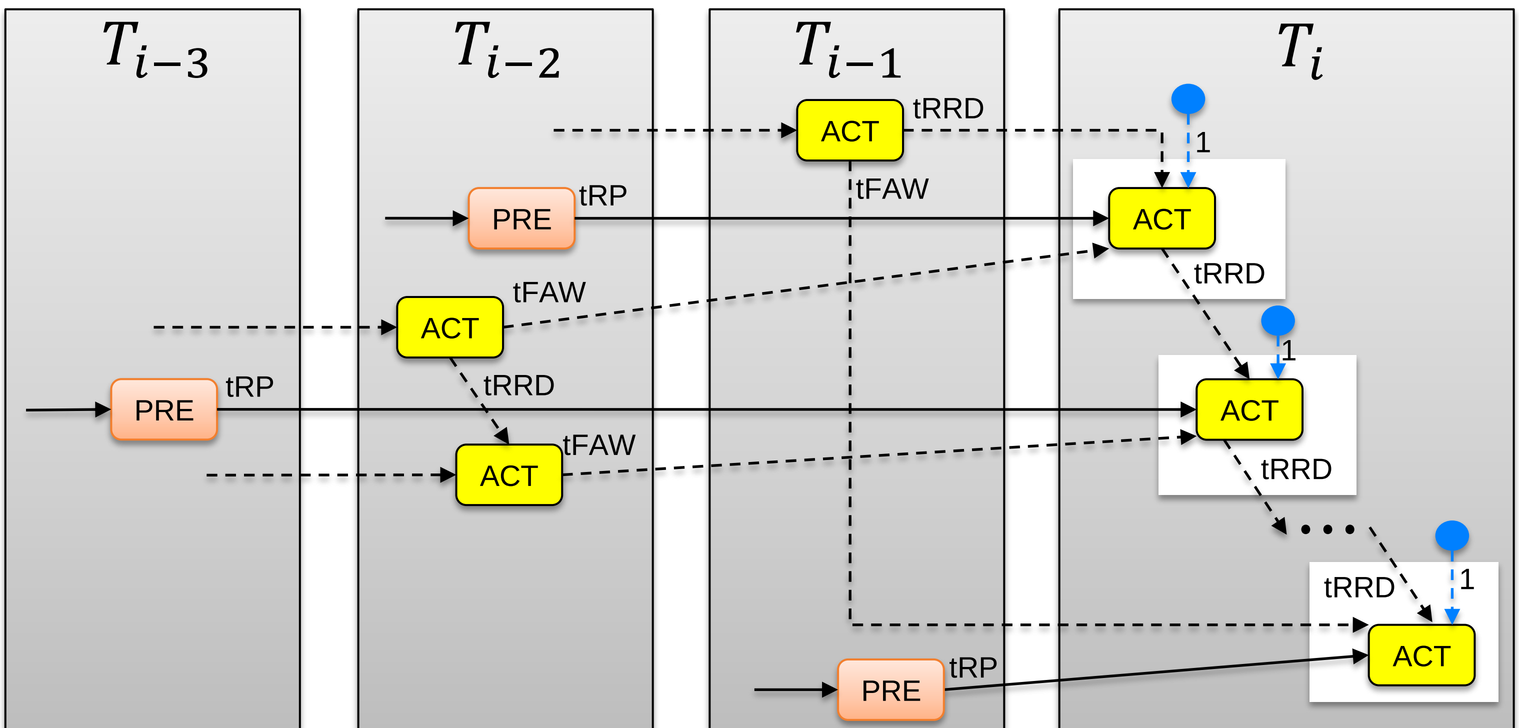


Outline

- Background
- Architecture and Command Scheduling Algorithm
- Formalization of Dynamic Command Scheduling
- WCET Analysis
- Experiments
- Conclusions

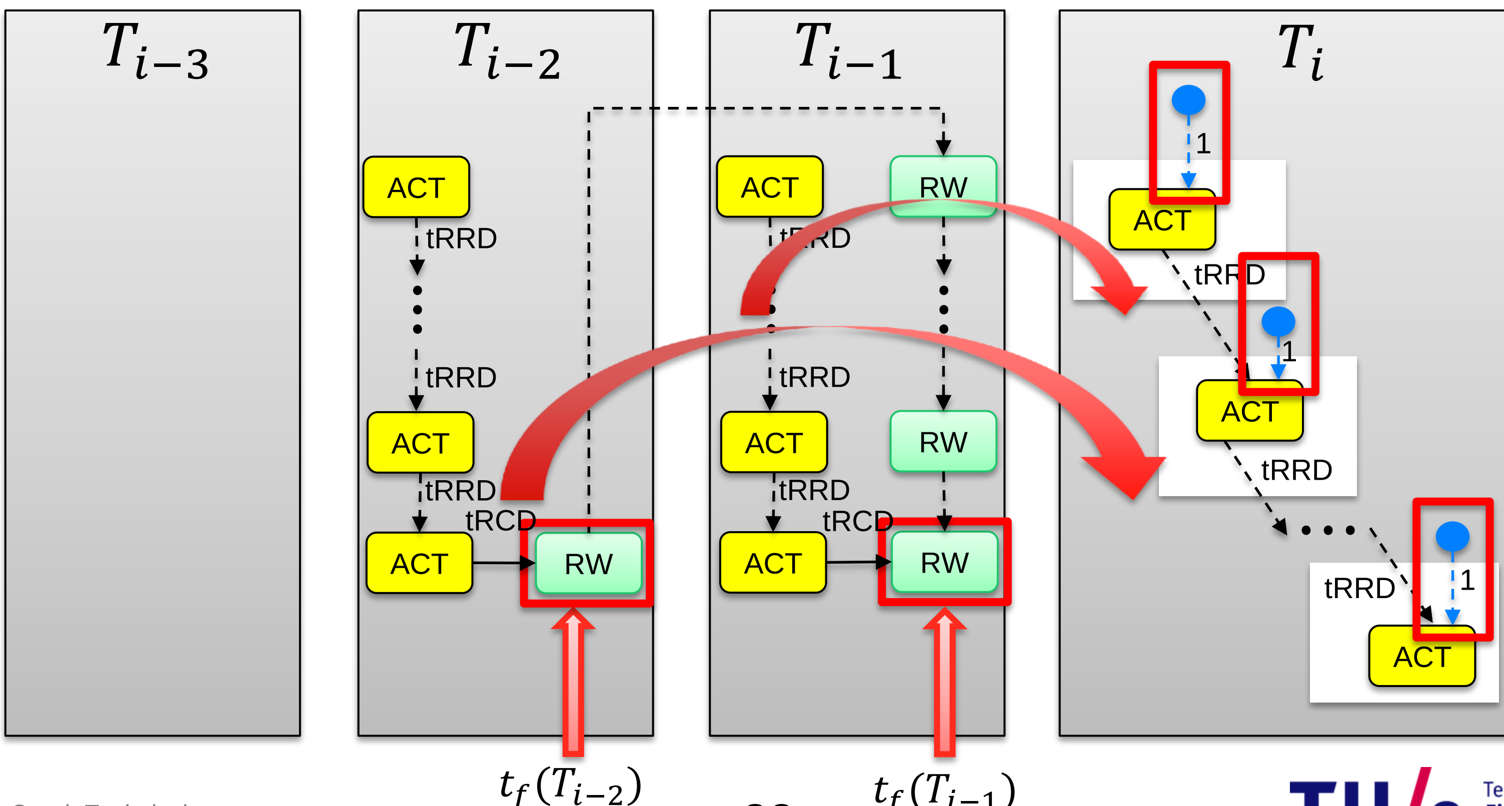
Worst-Case Finishing Time

- The maximum $t_f(T_i)$ is obtained by
  - maximizing the scheduling time of each ACT command



Worst-Case Finishing Time

- The maximum $t_f(T_i)$ is obtained by
  - maximizing the scheduling time of each ACT command
  - schedule commands of previous transactions as late as possible (ALAP) & assume a collision for each ACT



Theorem 1 (Variable transaction size)

- A transaction suffers WCET only if it starts with a bank that is the finishing bank of the previous write transaction

$$\hat{t}_f\left(T_i\right) = \max\{(BI_i \times BC_i - 1) \times tCCD, \\ (BI_i - 1) \times (tRRD + 1) + (BC_i - 1) \times tCCD\} \\ + t_f\left(T_{i-1}\right) + tRWTP + tRP + tRCD$$

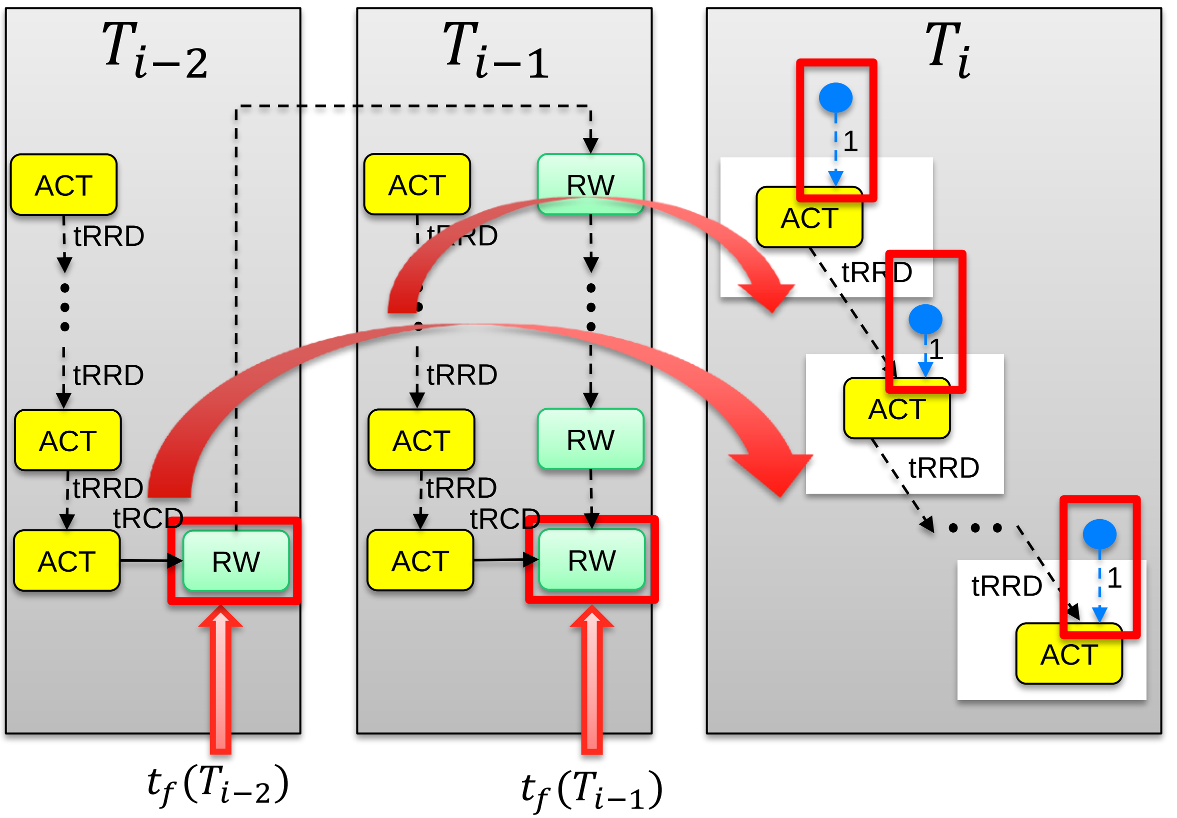
Theorem 2 (Fixed transaction size)

- With fixed size, a transaction suffers WCET only if the previous write transaction requires the same set of banks

$$
\begin{aligned}
\hat{t}_f(T_i) = t_f(T_{i-1}) + \max\{ & tRWTP + tRP + (BI \times BC - 1) \times tCCD \\
& - (BI - 1) \times \max\{tRRD, BC \times tCCD\} + tRCD \\
& + \max\{1, (BI - 1) \times (tRRD - BC \times tCCD) + BI\}, \\
& tSwitch + (BI \times BC - 1) \times tCCD\}
\end{aligned}
$$

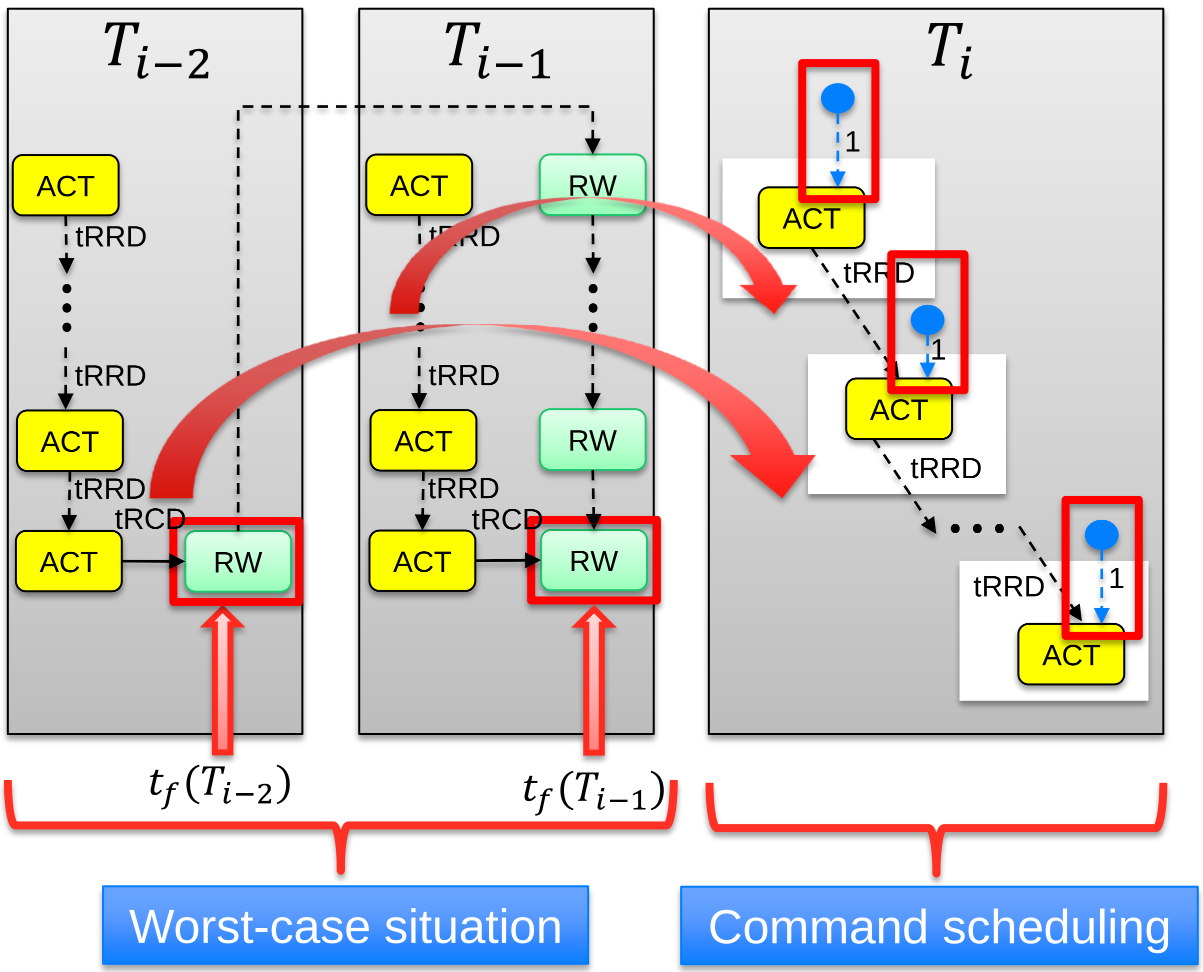
Worst-Case Finishing Time

- The analytical $\hat{t}_f(T_i)$ is pessimistic because of the conservative assumption of a collision for each ACT



Worst-Case Finishing Time (less pessimistic)

- Scheduled $\hat{t}_f(T_i)$ is given by a scheduling tool



Outline

- Background
- Architecture and command scheduling algorithm
- Formalization of dynamic command scheduling
- WCET analysis
- Experiments
- Conclusions

Experiments

- Goals
  - verify the validation of the formalization
  - for fixed/variable transaction sizes, respectively,
    - prove the execution time is upper bounded
    - show tightness of bound
    - obtain the average execution time

- Setup
  - cycle-accurate SystemC implementation
  - fixed-size transactions from Mediabench Application traces
  - variable-size transactions from synthetic traffic
  - 16bits DDR3-800/1600/2133 SDRAMs

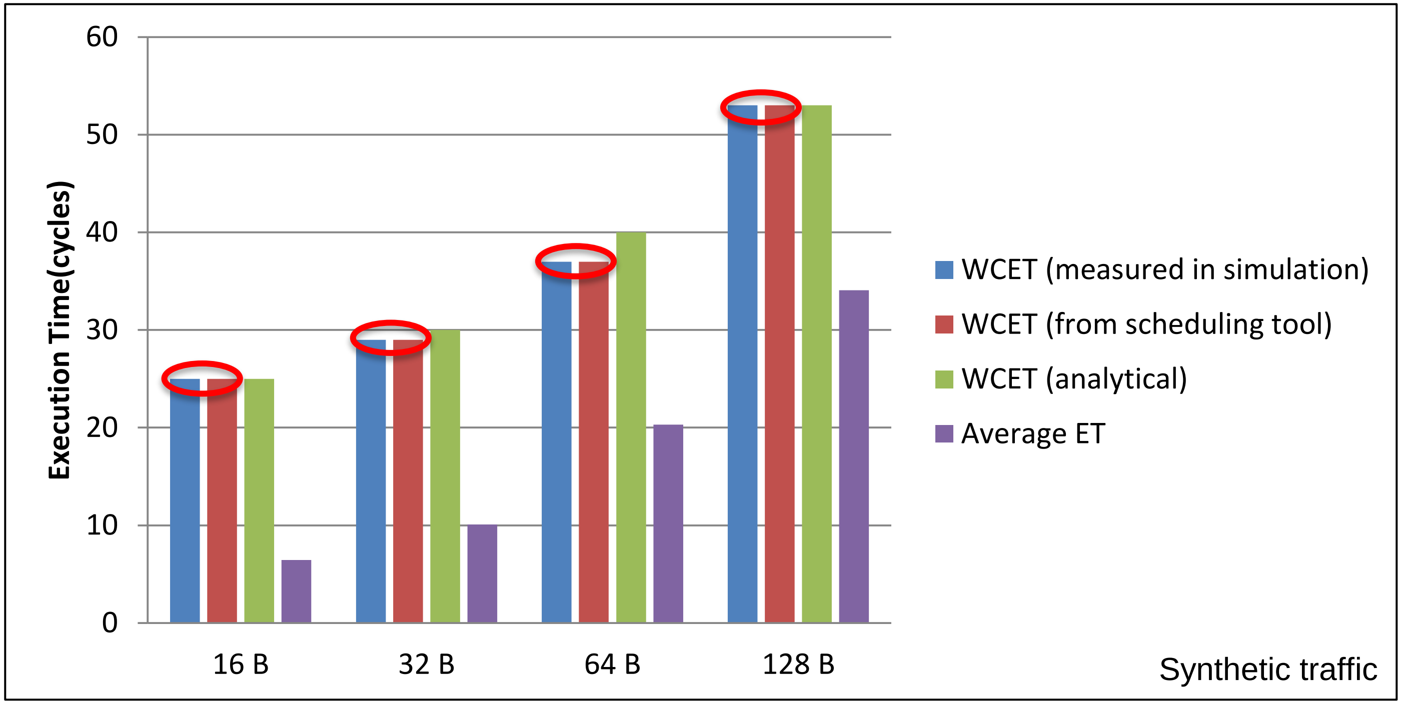
Experiment 1: Validation of Formalization

- The proposed formalism is implemented in C++ as an open source scheduling tool
  - *RTMemController*, http://www.es.ele.tue.nl/rtmemcontroller/

- The formalism accurately captures the SystemC implementation

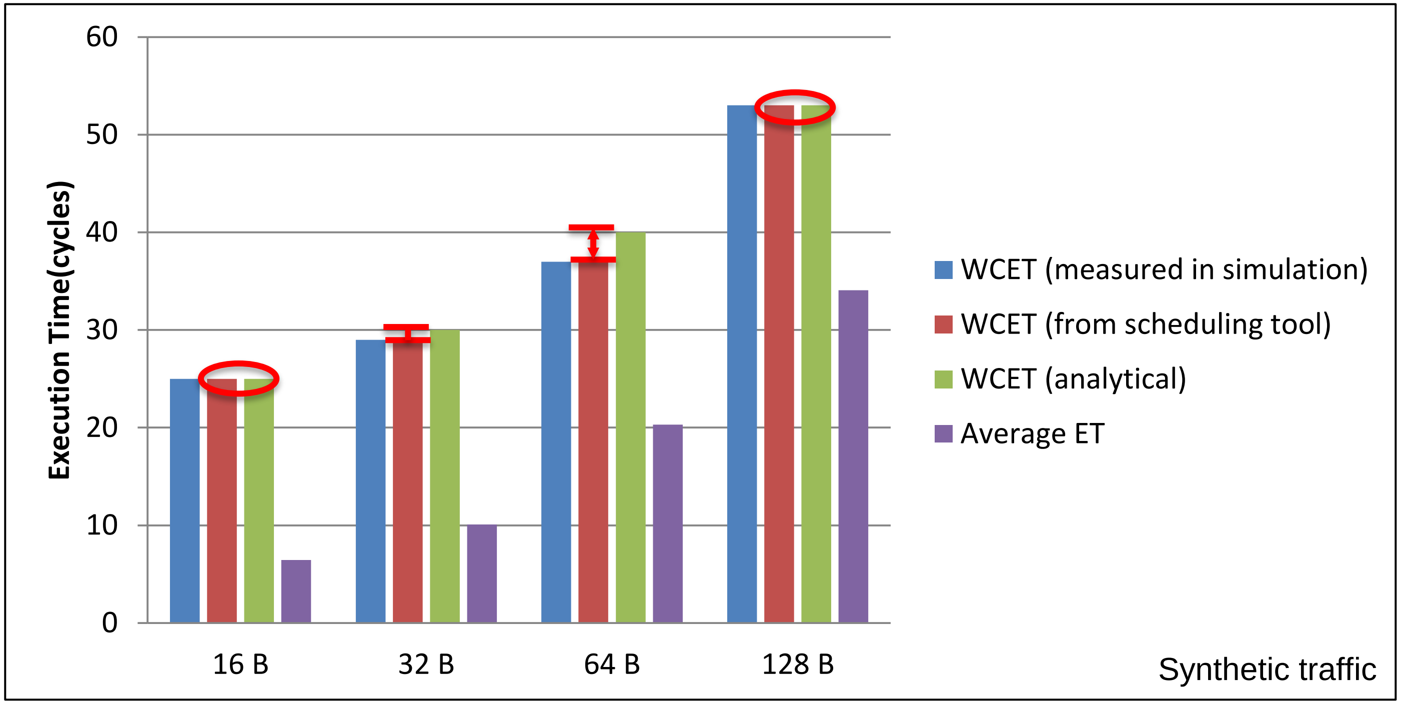
- It provides WCET and average ET results
  - the analytical and scheduled WCET
  - measured WCET

Experiment 2: Variable Transaction Size



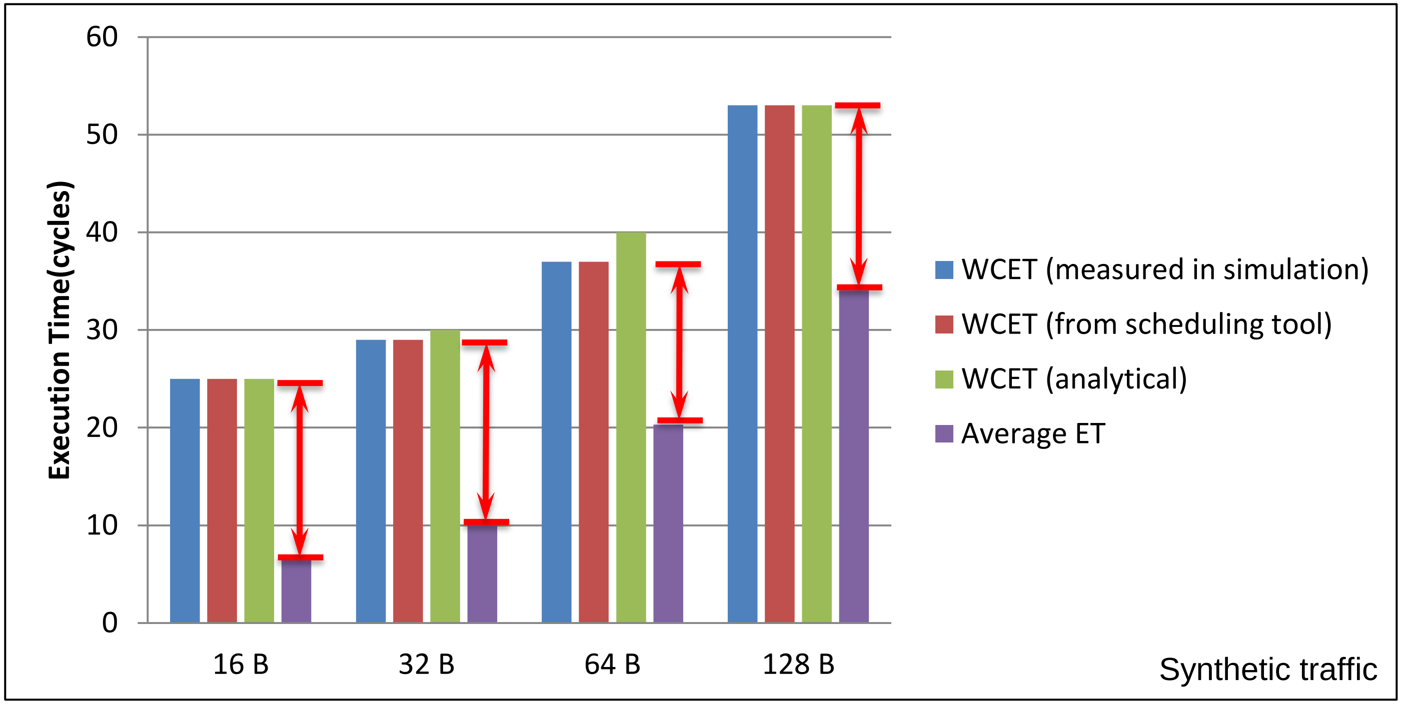
- The WCET bound is tight

Experiment 2: Variable Transaction Size



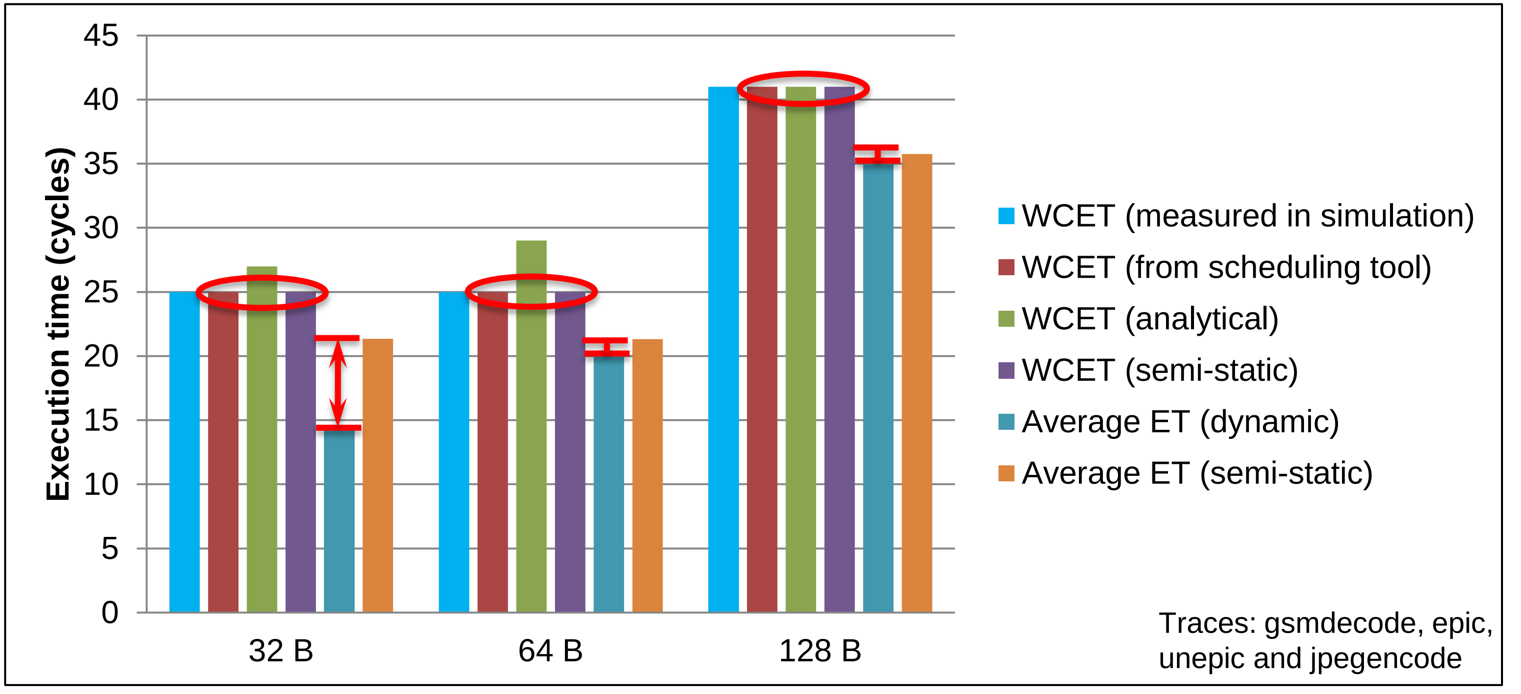
- Analytical WCET bound is pessimistic

Experiment 2: Variable Transaction Size



- Average ET is much lower than WCET (e.g., 74.4%)

Experiment 3: Fixed Transaction Size



- Compares to the semi-static approach
  - ➢ Better in average case (e.g., 38.6%), never worse in worst-case

Outline

- Background
- Architecture and command scheduling algorithm
- Formalization of dynamic command scheduling
- WCET analysis
- Experiments
- Conclusions

Conclusions

- A back-end architecture with a scheduling algorithm for dynamic command scheduling
- Valid formalization & analysis of WCET
- *RTMemController:* an open source scheduling tool based on the formalism and provides both scheduled & analytical WCET, and average ET
- WCET bound is tight
- Dynamic scheduling outperforms the semi-static approach in the average case (max. 38.6%) while performing at least equally well in the worst-case

Thank You.

yonghui.li@tue.nl

RTMemController: http://www.es.ele.tue.nl/rtmemcontroller/