Timing Anomalies in Multi-core Architectures due to the Interference on the Shared Resources

Hardik Shah, Kai Huang and Alois Knoll

Robotics and Embedded Systems

Department of Informatics

Technische Universität München

www6.in.tum.de

23 January 2014

Define: Anomaly

- **Dictionary:**
  - a deviation from the common rule, type, arrangement, or form.
  - Synonyms: abnormality, exception, peculiarity, irregularity.

Anomaly: Counter intuitive behavior

Timing anomaly: Counter intuitive timing behavior

- The term was first coined by Lundqvist & Stenström [1].
- Analysis and Modeling: [1], [2], [3], [4], [12], [13], [14], [15], [16].
- Real-life examples are missing

Motivation

- **Multi-cores everywhere:**
  - Demanding real-time applications.
  - Only multi-cores will be produced in future !
- **Interference on shared resources, e.g. shared memory is the biggest challenge for the WCET analysis of applications executing on multi-core architectures. The interference analysis must be done carefully.**

Motivation

- **Interference:**
  - Occurs deep inside chip and is invisible outside.
  - Depends on applications executing on co-existing cores.
  - Why not measure execution time of application-under-test in the presence of aggressive co-existing applications and consider the maximum execution time as WCET? [18], [19], [20]
- **Contribution: Identified two new timing anomalies:**
  - Occurs due to the interference on shared resources.
  - Real-life examples using MälardalenWcet benchmark suit and NIOS II quad-core processor on Altera FPGA.

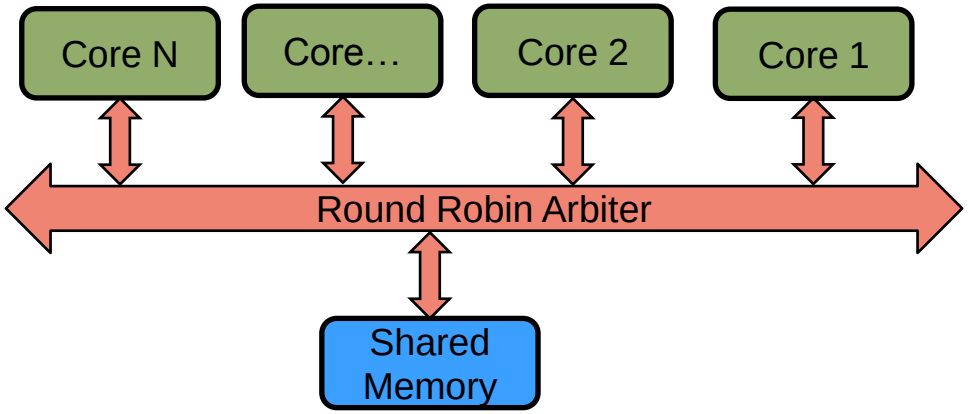
Agenda

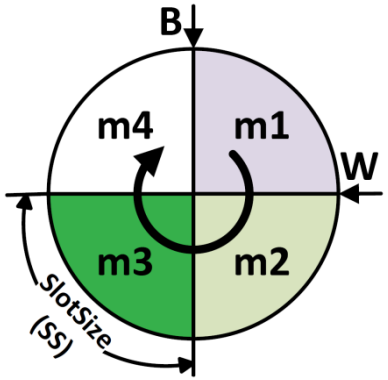
- Related work
- Background
- Theory behind the anomalies
- Real-life examples
- Conclusion

Related work

- **Interference analysis:**
  - Cognizant approach:
    - Takes cognizance of the co-existing applications.
    - Lv et al [7], Pellizonni et al [8]
  - Isolation approach:
    - Considers the worst possible interference
    - Our previous work [9, 10, 11], Paolieri et al [12]
- **Timing anomalies:**
  - [13] models timing anomalous processor
  - [14] identifies a new timing anomaly
  - [15, 16] analyzes WCET on timing anomalous multi-core architectures

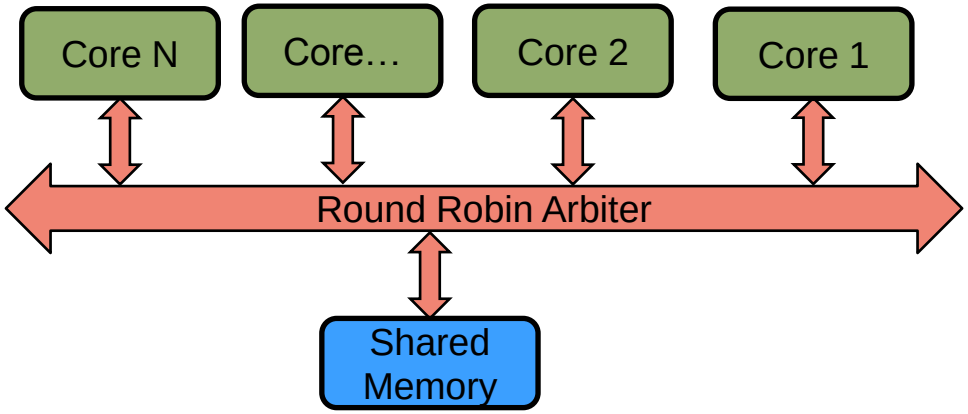
Background: Round robin arbiter (Greedy TDMA)

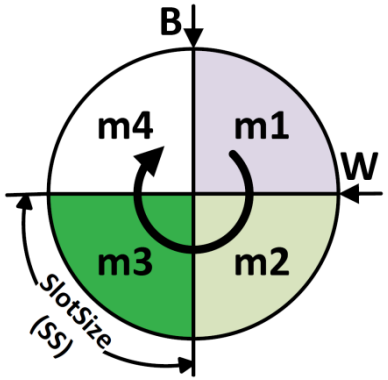




- **Application-under-test executes on m1 (core1).**
- **The arbiter continuously looks for an active master in the clockwise direction.**
  - As soon as an active master is encountered, it is granted access to the shared resource for `SlotSize` number of clock cycles.
- **Work conserving.**

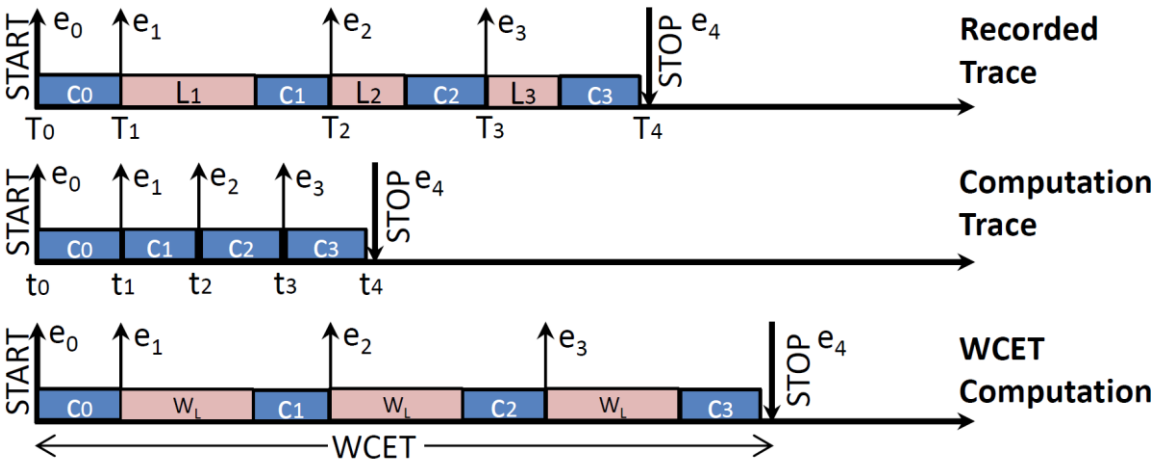
Background: Round robin arbiter





- **Best case completion latency, B_L = 1 x SS, m1 issues a request when the arbiter pointer is at B.**
- **Worst case completion latency, W_L = 4 x SS, m1 issues a request when the arbiter pointer is at W and ALL other masters utilize their slots. $A_L = (B_L + W_L)/2$.**

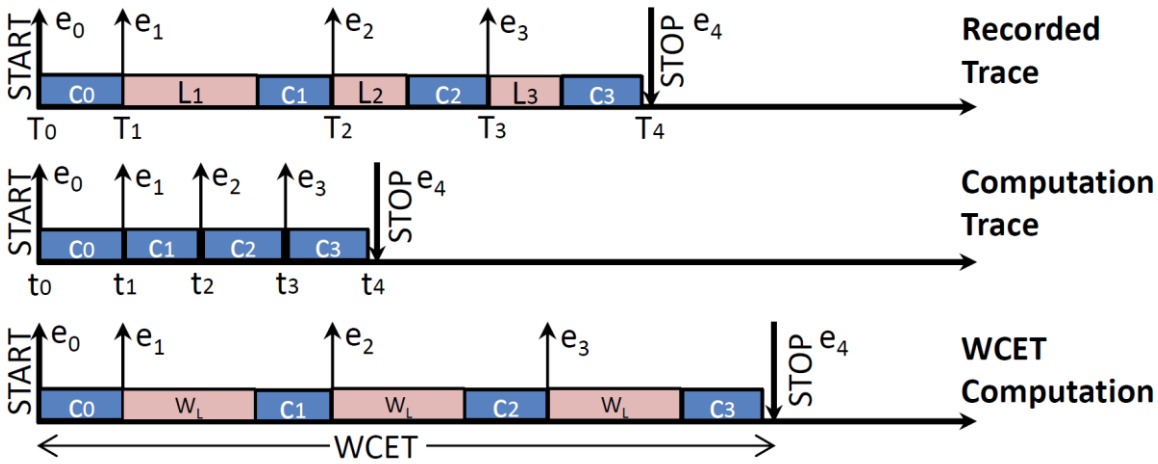
Background: Computation trace



W_L = Worst case latency, L_x = Measured latency, c_x = Computation time,
e_x = x^{th} event, T_x = Time in recorded trace, t_x = Time in computation trace

- **Recorded trace is extracted using an ISS.**
- **Cache misses are denoted by e_x events.**
- **c_x is the time between issue of two consecutive cache misses.**
  - **During c_x, processor executes from caches and registers.**
- **Experienced latencies $L_x \in [B_L, W_L]$.**

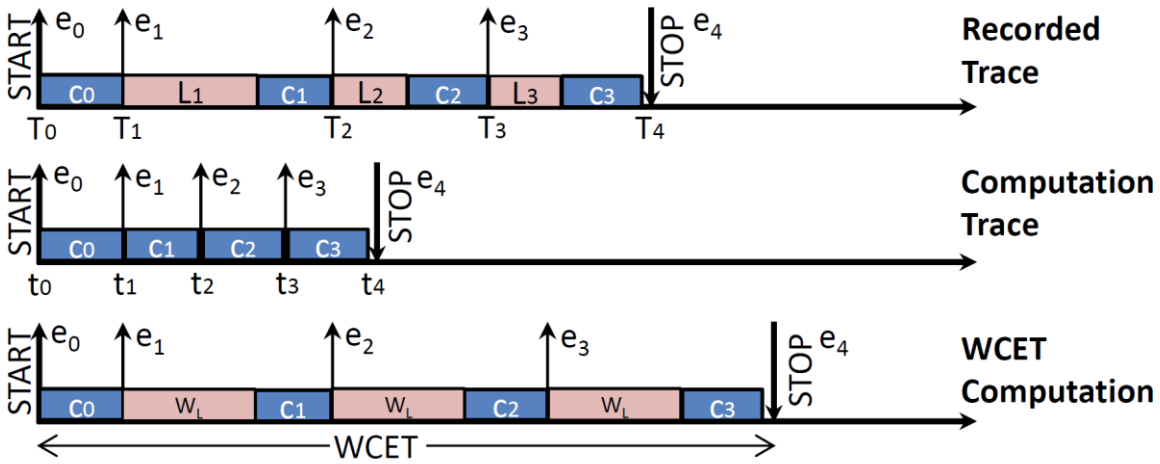
Background: Computation trace



W_L = Worst case latency, L_x = Measured latency, c_x = Computation time,
e_x = x^{th} event, T_x = Time in recorded trace, t_x = Time in computation trace

- **Experienced latencies are removed in computation trace and all cache miss events are shifted to the left in time.**
- **Each event is appended by W_L and they are shifted to right in time to consider the worst case interference.**

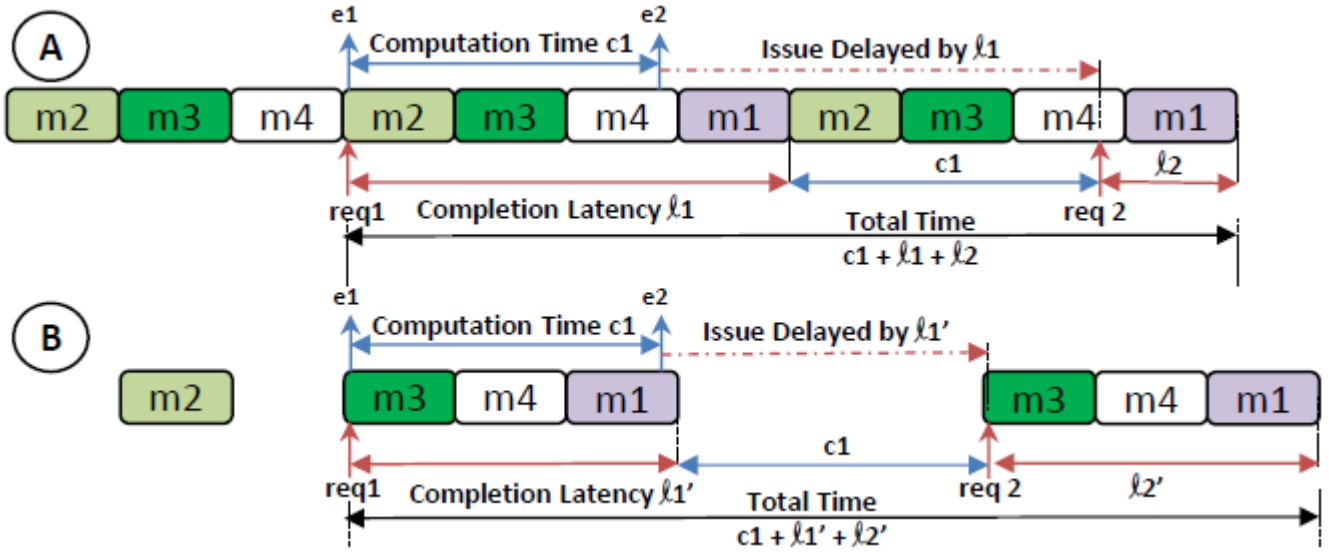
Background: Computation trace



W_L = Worst case latency, L_x = Measured latency, c_x = Computation time,
e_x = x^{th} event, T_x = Time in recorded trace, t_x = Time in computation trace

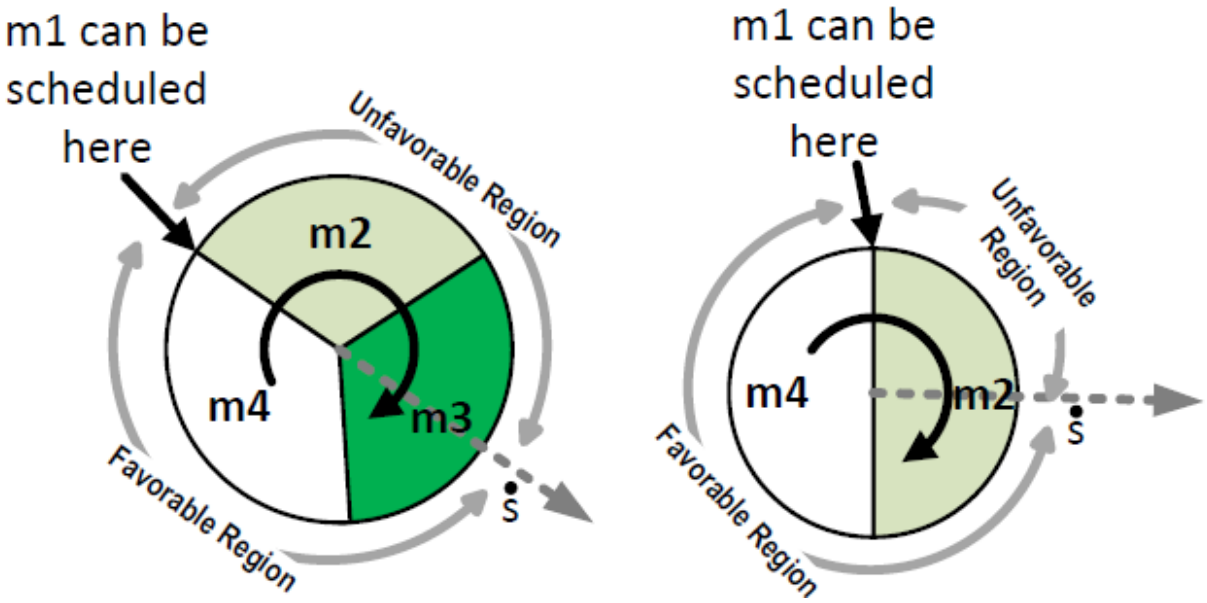
- **The computation trace can be considered constant if we start from the same cache, pipeline state and use the same input data.**

Background: Latencies under the round robin arbitration



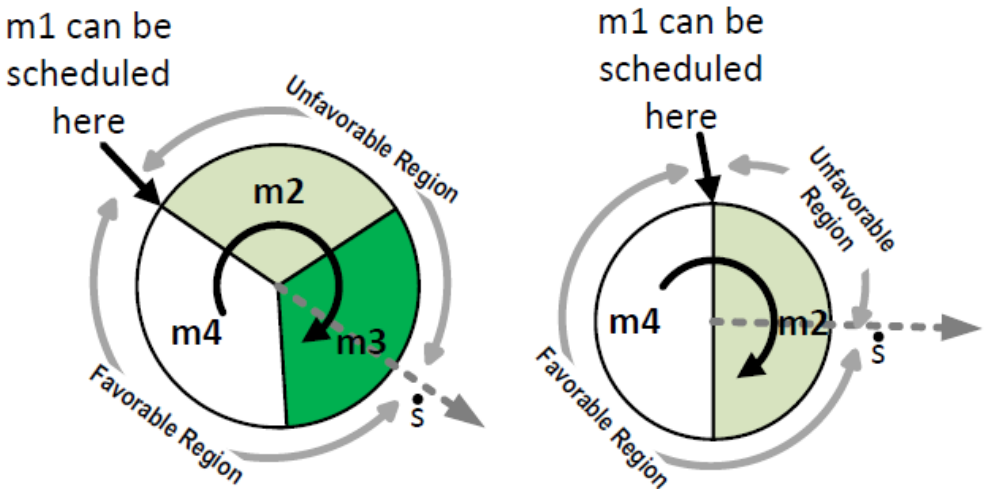
- **Using computation trace, different interference scenarios can be assumed.**
- **The completion latency experienced by one event delays the issue of the next event by the same amount.**
  - Considering single outstanding cache miss.

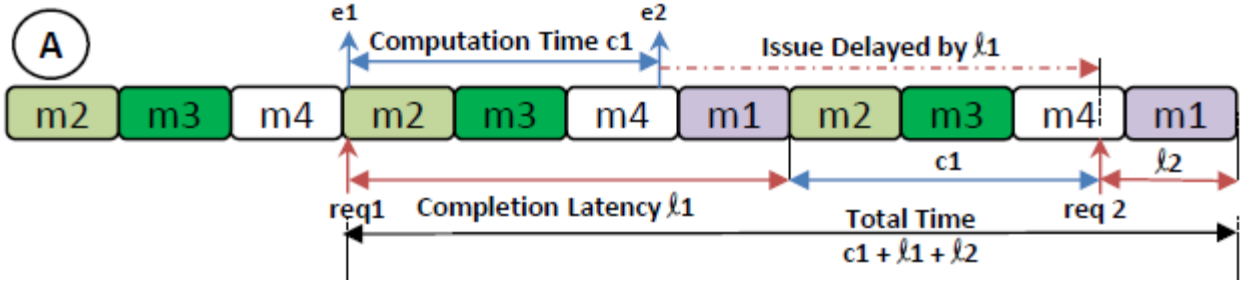
α - Interference



- **Definition: Uninterrupted interference generated from α number of co-existing masters.**
  - Any co-existing master either interferes uninterruptedly or it is inactive.
- **Occurs many times during application execution.**
  - After reset, after new task is scheduled on co-existing core, memory intensive co-existing applications, e.g. camera, radar etc.

Latency under α - Interference

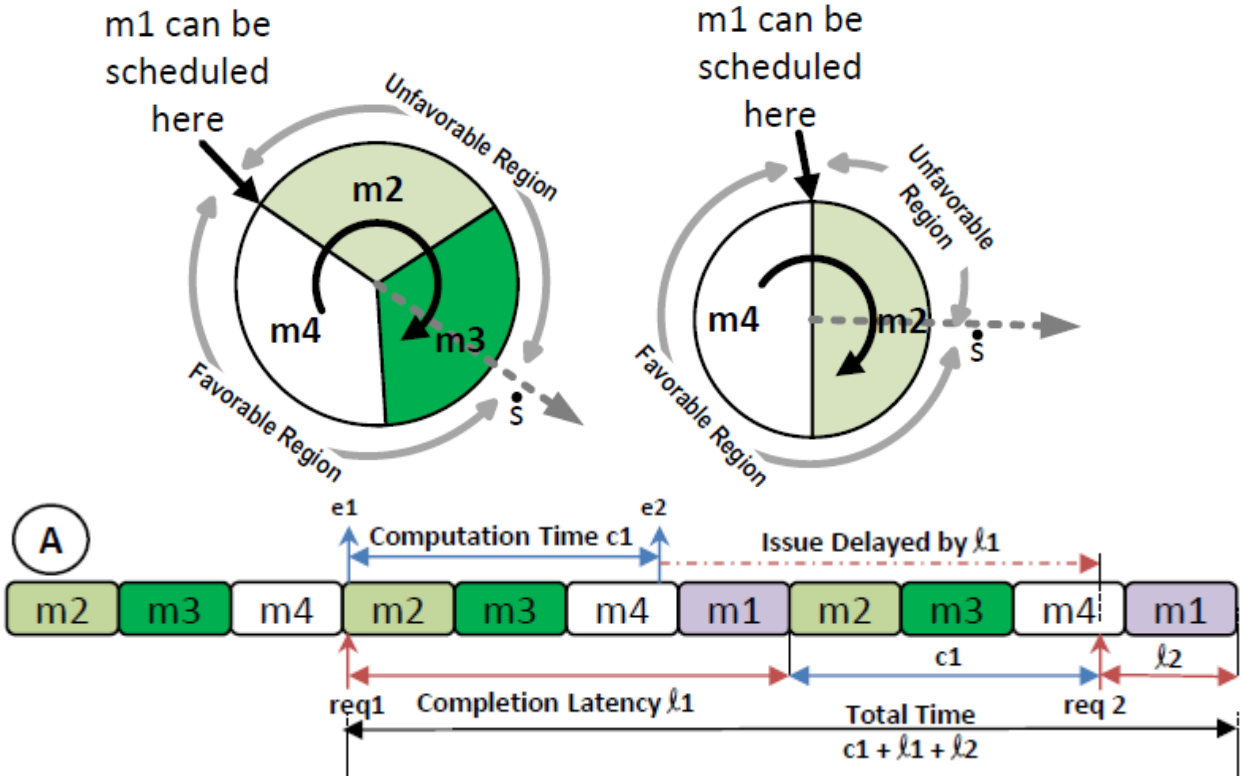




$$DL_{\alpha}^{i} = (\alpha + 1) \times SS - \{c_{(i-1)} \bmod (\alpha \times SS)\}$$

The arbiter pointer rotation becomes deterministic.

Latency under α - Interference



$$DL_{\alpha}^{i} = (\alpha + 1) \times SS - \{c_{(i-1)} \bmod (\alpha \times SS)\}$$

$$DL_{\alpha}^{i} = (\alpha + 1) \times SS - \Theta_{\alpha}^{(i-1)}$$

Latency under α - Interference

- **For an individual (i^{th}) access:**

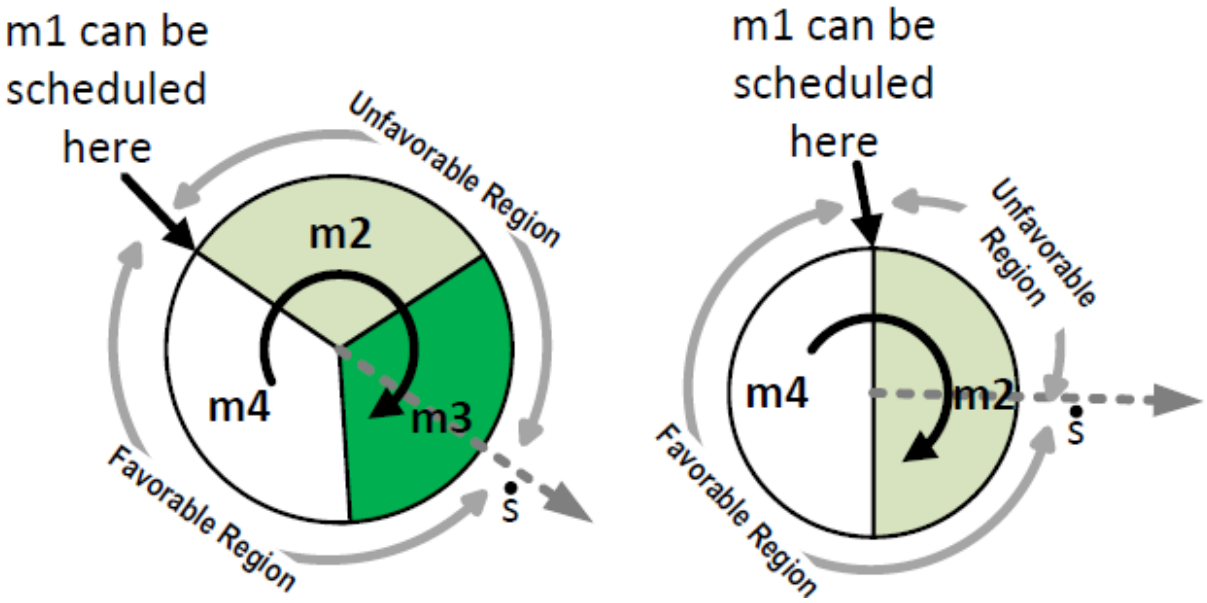
$$DL_{\alpha}^{i} = (\alpha + 1) \times SS - \Theta_{\alpha}^{(i-1)}$$

- **Considering average value of all accesses along the execution path:**

$$\overline{DL_{\alpha}} = (\alpha + 1) \times SS - \overline{\Theta_{\alpha}}$$

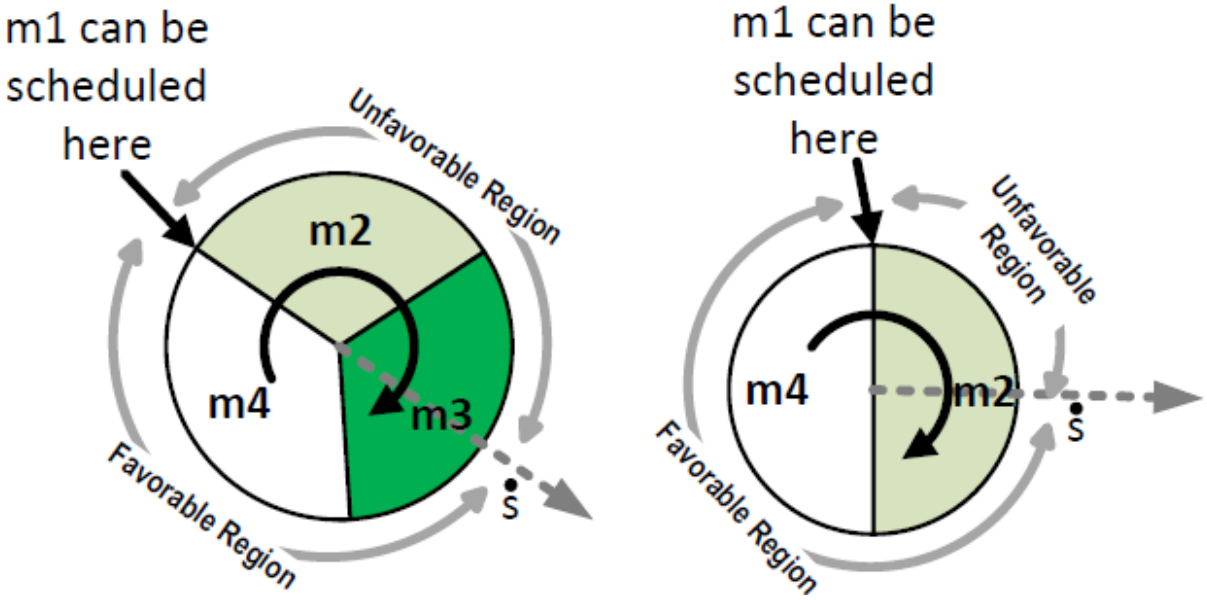
- $\overline{DL_{\alpha}}$ **is an important parameter in determining the average experienced latency by an application execution path under α interference.**

Timing anomaly - 1



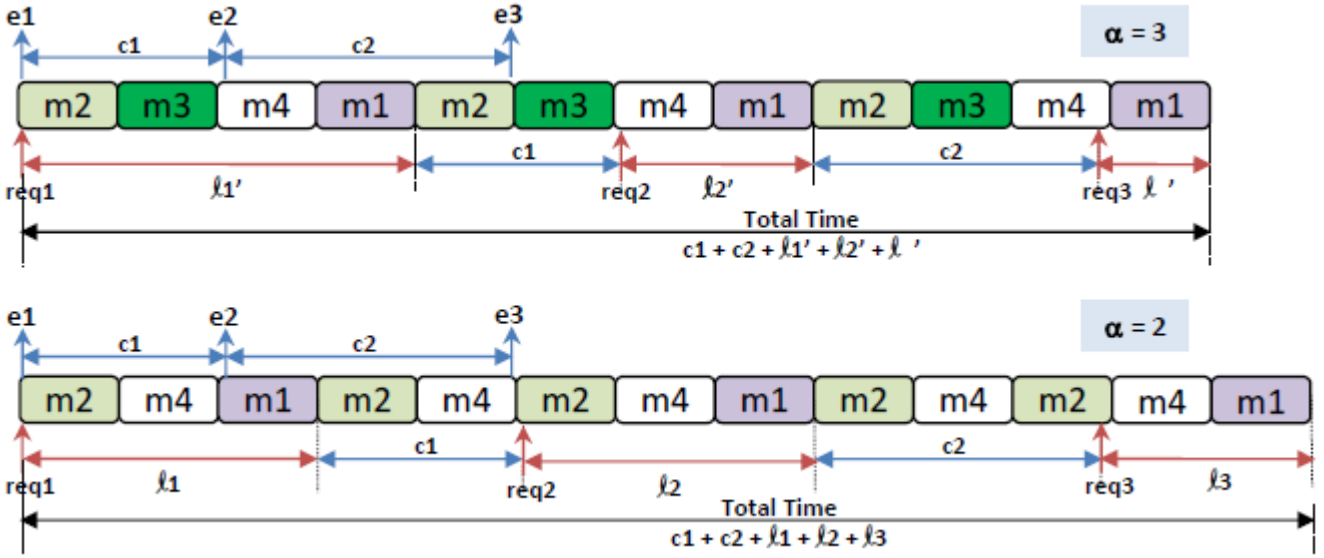
- **The round robin wheel is divided in,**
  - Favorable region: $L_X < A_L$.
  - Unfavorable region: $L_X > A_L$.

Timing anomaly - 1



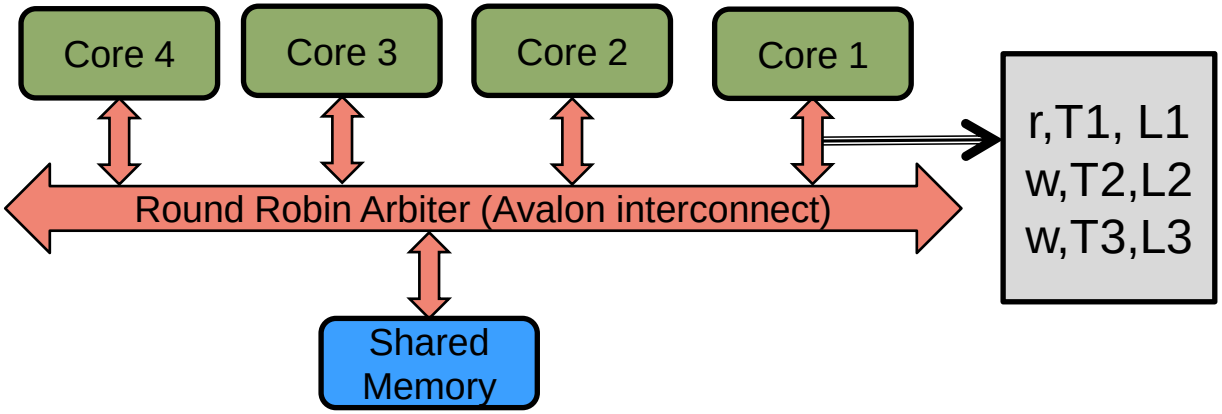
- **Application which does majority of accesses in the favorable region, benefits from the uninterrupted interference and the experienced average latency is less than the theoretical average-case latency (A_L).**

Timing anomaly - 2



- **For some applications, uninterrupted interference from less number of masters produce longer latencies than more number of masters.**

Test setup



- **Goal:**
  - Explore real-life examples of the anomalies.
- **Altera Quad-core NIOS processor.**
  - **32 Bytes cache line-size.**
- **Mälardalen WCET benchmark suit.**
- **Trace capture using Altera cycle accurate simulators.**

Test 1

These two applications experienced less than average case execution time under $\alpha = 3$ interference

Benchmark	OET	ACET	WCET	DL_3
cover	12207	11586	14274	24.95
crc	104331	101196	108396	26.57
duff	6777	5936	7364	28.63
edn	**360574**	**361342**	**391834**	**19.91**
expint	16573	16469	16781	23
fac	1129	1107	1227	24
fdct	**22079**	**24173**	**32453**	**18**
fibcall	1110	1098	1182	24
jane	832	813	921	25
jfdcint	28209	28107	33891	21.97
minver	158910	142289	189893	25.95
prime	196676	186533	228197	25
quart	224508	204366	271530	24.99
ud	38248	34815	46443	24.93

Execution times in clock cycles. α = 3. OET = Observed Execution Time, ACET = Average Case Execution Time

- **Altera Quad-core NIOS processor.**
  - 32 B cache line-size, I$ & D$ size = 512 B.
  - ACET and WCET are achieved by appending cache miss events by the theoretical average-case latency A_L and the theoretical worst case latency W_L, respectively.

Test 2

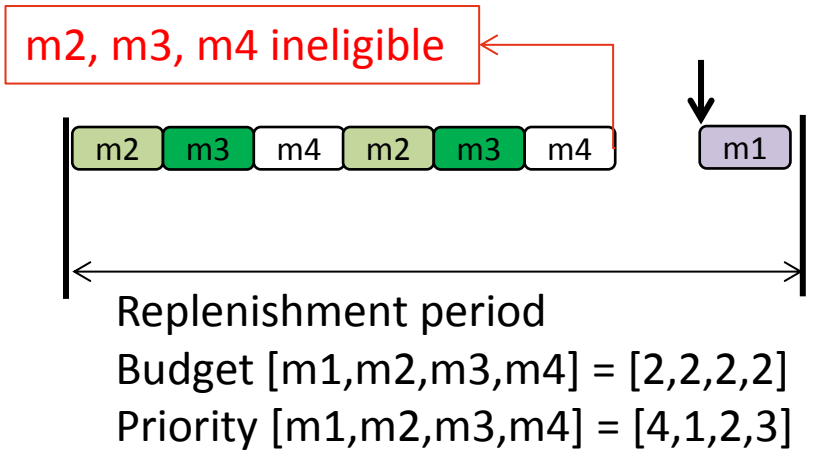
The cover application experienced more latencies under $\alpha = 4$ interference than $\alpha = 5$ interference.

	α	$\alpha = 3$	$\boldsymbol{\alpha = 4}$	$\boldsymbol{\alpha = 5}$	$\alpha = 6$	$\alpha = 7$
cover	OET	10717	**11686**	**11046**	11942	13662
	$\overline{DL}$	23	**30**	**27**	32	42

OET in clock cycles

- **Core configuration**
  - 32 B cache line-size, I$ & D$ size = 1024 B.
- **Starting from a quad-core system ($\alpha = 3$), we kept on increasing number of cores to 8 ($\alpha = 7$).**

Discussion



- **Round robin arbiter is popular and default arbiter of the many off-the-shelf architectures, e.g. Altera, LEON etc.**
- **The first timing anomaly is also observed under advanced budget based arbiters, e.g. CCSP [11], PBS [9].**
  - Budgeted number of transfers per unit time.
  - Conflict resolution by priorities.
- **Both the timing anomalies are absent under TDMA and Priority Division [21] arbiters.**

Conclusion

- **Identified two new timing anomalies which occur due to shared resource interference in multi-core architectures.**
  - Some applications benefit from aggressive co-existing applications and experience even less than the average-case latencies.
  - Some applications experience more latencies in the presence of less number of aggressive interfering applications than in the presence of more number of aggressive interfering applications.
- **The real-life examples of the presence of the timing anomalies are presented using Mälardalen WCET benchmark suit and multi-core processor implemented on an Altera FPGA.**

Thank you for your attention

Questions ?