Bounding DRAM Interference in COTS Heterogeneous MPSoCs for Mixed Criticality Systems

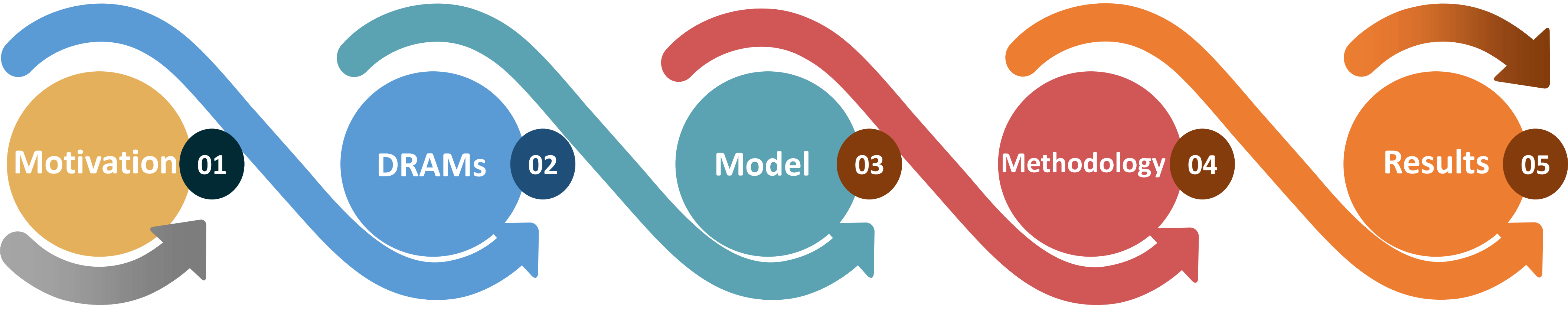
Mohamed Hassan and Rodolfo Pellizzoni

UNIVERSITY of GUELPH

intel

UNIVERSITY OF WATERLOO

EMBEDDED SYSTEMS WEEK



Motivation 01

DRAMs 02

Model 03

Methodology 04

Results 05

Outline

- **Emerging Systems No longer solely hosting isolated safety-critical tasks**
  - Execute tasks with different criticalities

Bounding DRAM Interference in COTS Heterogeneous MPSoCs for Mixed Criticality Systems

(ABS)

- Engine Control Unit (ECU)

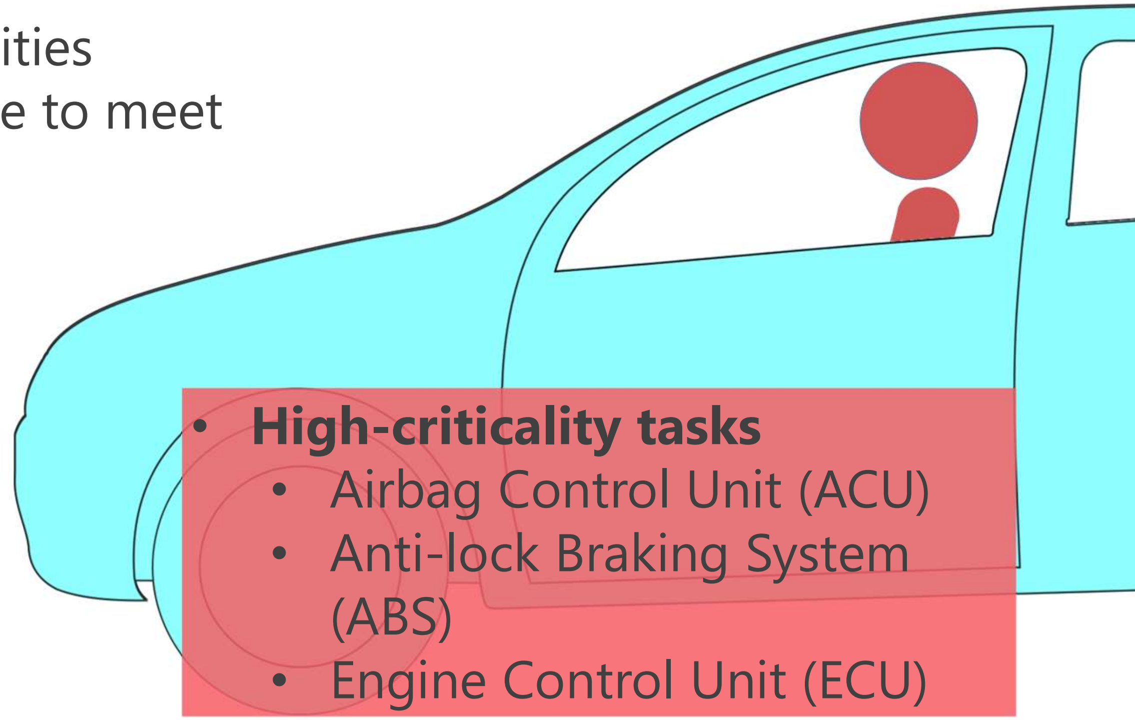
Mixed Criticality Systems

MOTIVATION

Mixed Criticality Systems

MOTIVATION

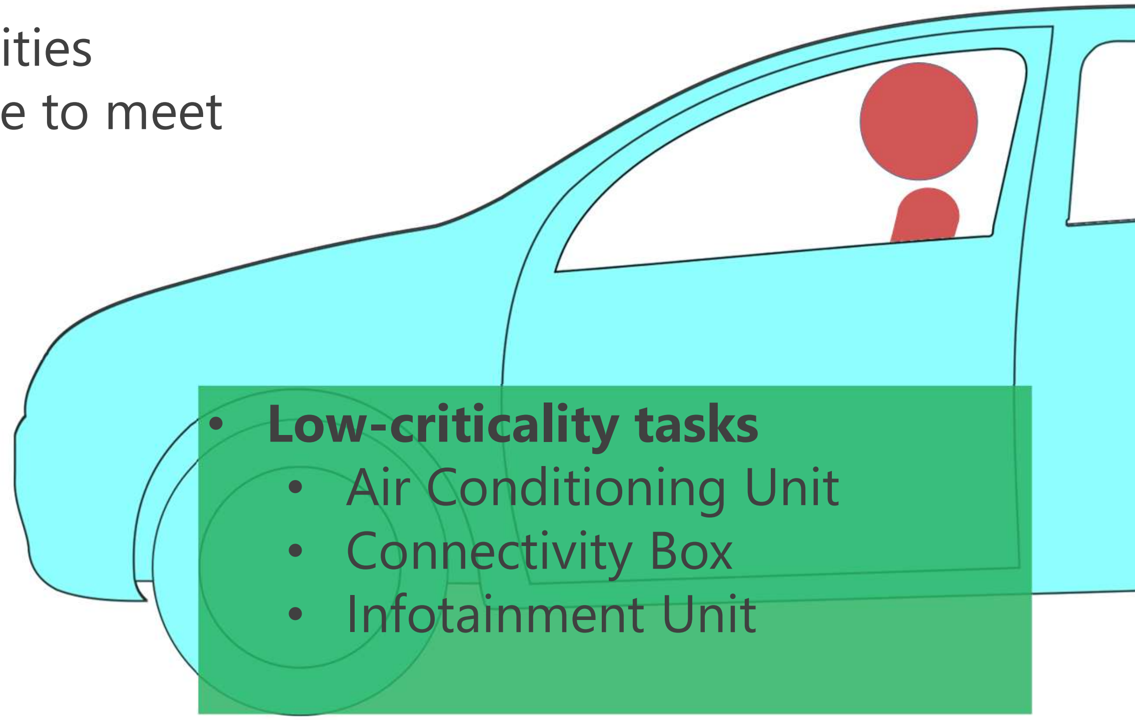
- **Emerging Systems No longer solely hosting isolated safety-critical tasks**
  - Execute tasks with different criticalities
  - Criticality α consequences of failure to meet requirements



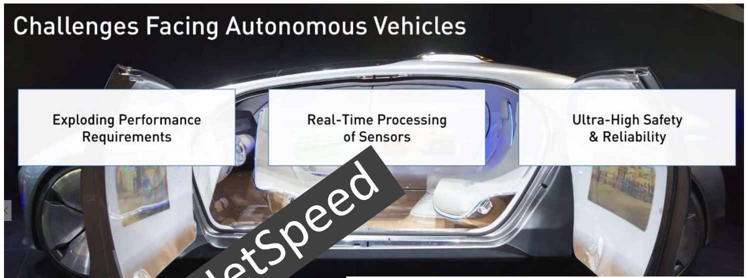
- **High-criticality tasks**
  - Airbag Control Unit (ACU)
  - Anti-lock Braking System (ABS)
  - Engine Control Unit (ECU)

Mixed Criticality Systems

- **Emerging Systems No longer solely hosting isolated safety-critical tasks**
  - Execute tasks with different criticalities
  - Criticality α consequences of failure to meet requirements



- **Low-criticality tasks**
  - Air Conditioning Unit
  - Connectivity Box
  - Infotainment Unit



Challenges Facing Autonomous Vehicles

Exploding Performance Requirements

Real-Time Processing of Sensors

Ultra-High Safety & Reliability

NetSpeed

Key Requirements of Automotive-Grade IP

Reduce Risk and Accelerate Qualification for Automotive SoCs

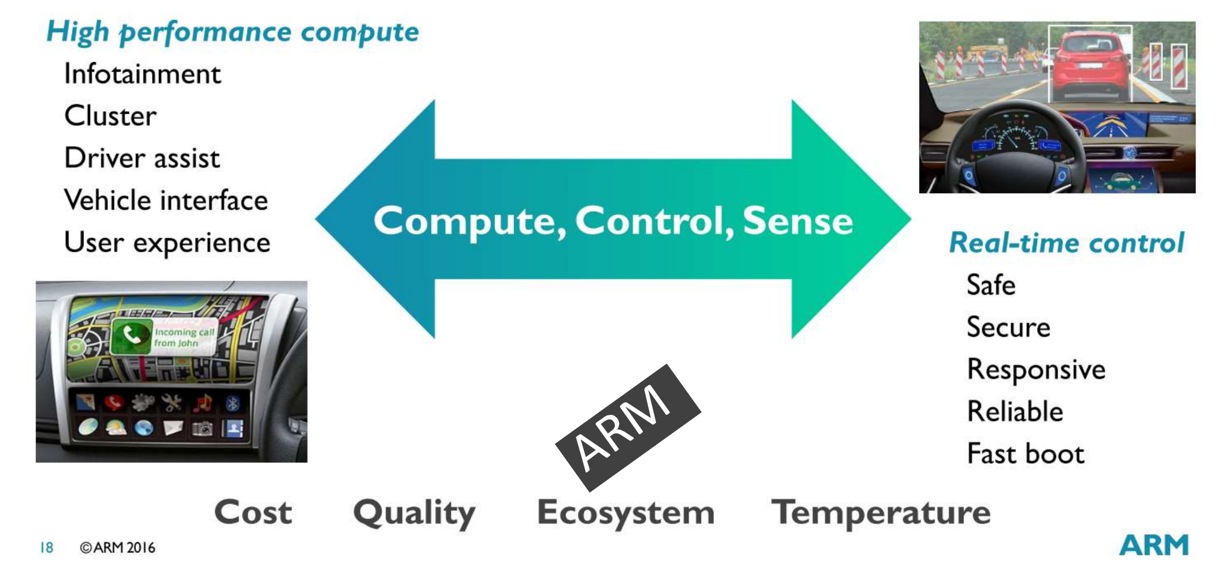
3

✓ **Functional Safety** — Accelerate ISO 26262 functional safety assessments to help ensure designers reach target ASIL levels

✓ **Reliability** — Reduce risk & develop AEC-Q100 qualific

for automotive applications

Synopsys



High performance compute

- Infotainment
- Cluster
- Driver assist
- Vehicle interface
- User experience

Compute, Control, Sense

Real-time control

- Safe
- Secure
- Responsive
- Reliable
- Fast boot

Cost **Quality** **Ecosystem** **Temperature**

ARM

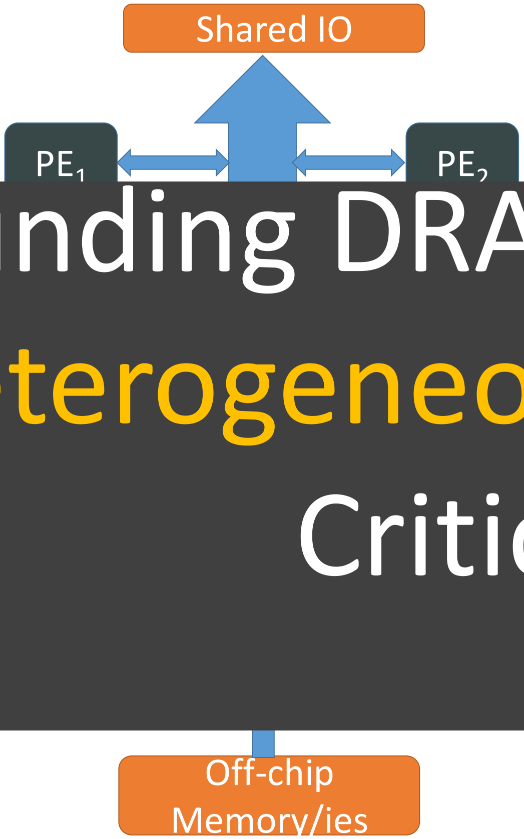
18 ©ARM 2016

ARM

Mixed Criticality Systems

MOTIVATION

Bounding DRAM Interference in COTS Heterogeneous MPSoCs for Mixed Criticality Systems

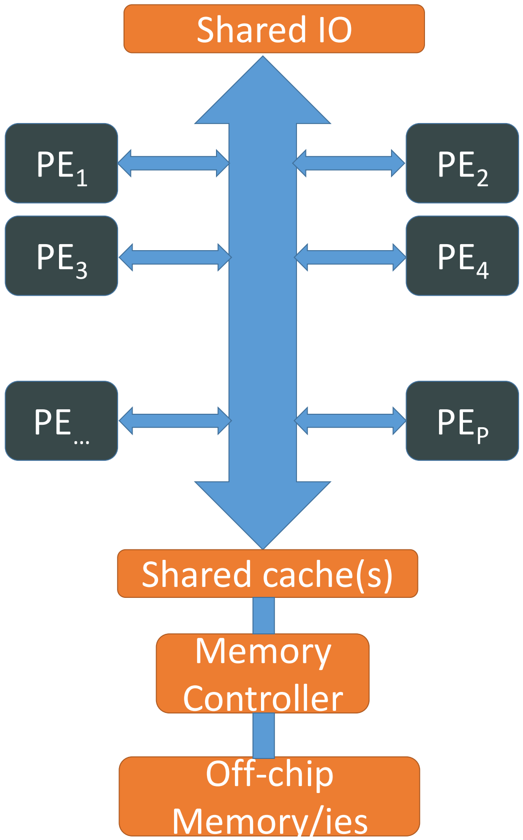


MPSoCs

MOTIVATION

MPSoCs

MOTIVATION



Shared IO

PE_1

PE_2

PE_3

PE_4

$PE_{...}$

PE_P

Shared cache(s)

Memory Controller

Off-chip Memory/ies

Why MPSoCs?

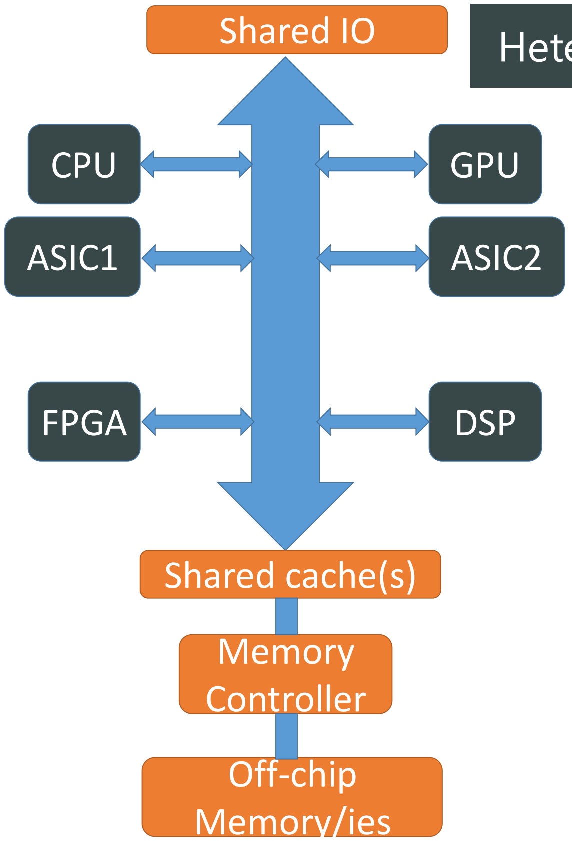
- Low cost
- High performance
- Energy Efficiency
- Low time-to-market (3rd party IPs)

Why Heterogenous MPSoCs?

- Variety of processing capabilities

→ Best-suits MCS conflicting requirements

Heterogenous MPSoCs



Heterogenous MPSoCs

MOTIVATION

Complementary SoC processor requirements

High performance compute

Infotainment
Cluster
Driver assist
Vehicle interface
User experience

Compute, Control, Sense

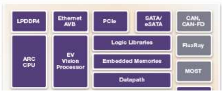
Real-time control

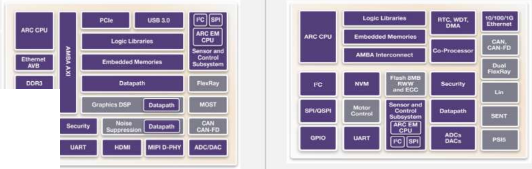
Cost **Quality** **Ecosystem**

Automotive Applications Require Different SoC Architectures

5

High-End ADAS **Infotainment** **MCU**



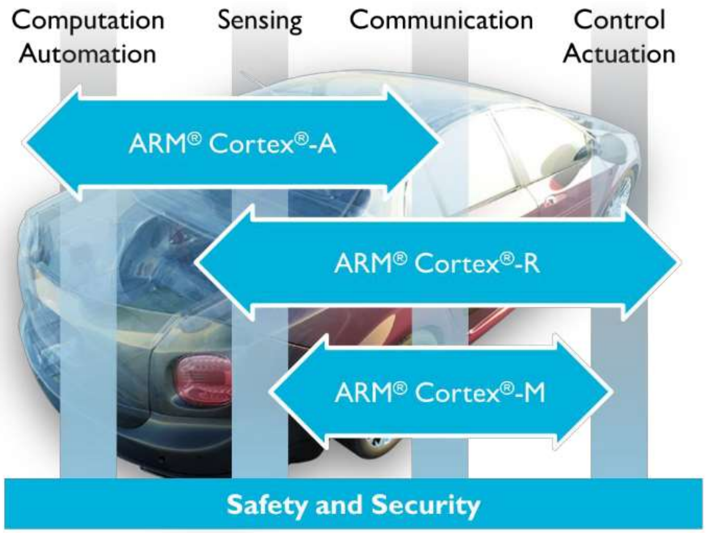


)R4, Ethernet AVB, MIPI, HDMI,
, ADC, UFS, eMMC
Multimedia

ion

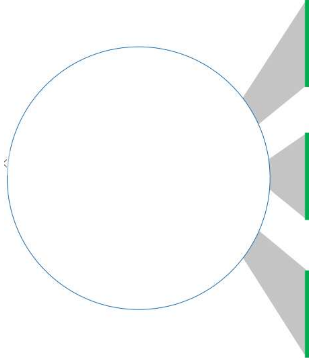
- IP: Ethernet 10/100/1000, ADC, I/F peripherals
- Medium Density NVM

SYNOPSYS®



ARM

Translating System-Level Requirement



Exploding Performance Requirements	Rise of het Cache coh
Real-Time Sensor Processing	Different I Ensuring c
Ultra-High Safety & Reliability	Pressure to comply to industry standards – ISO 26262 Functional Safety – Performance – Area Tradeoffs

omputing

- Smaller amounts of data
- Highly structured data
- Complex computation/item

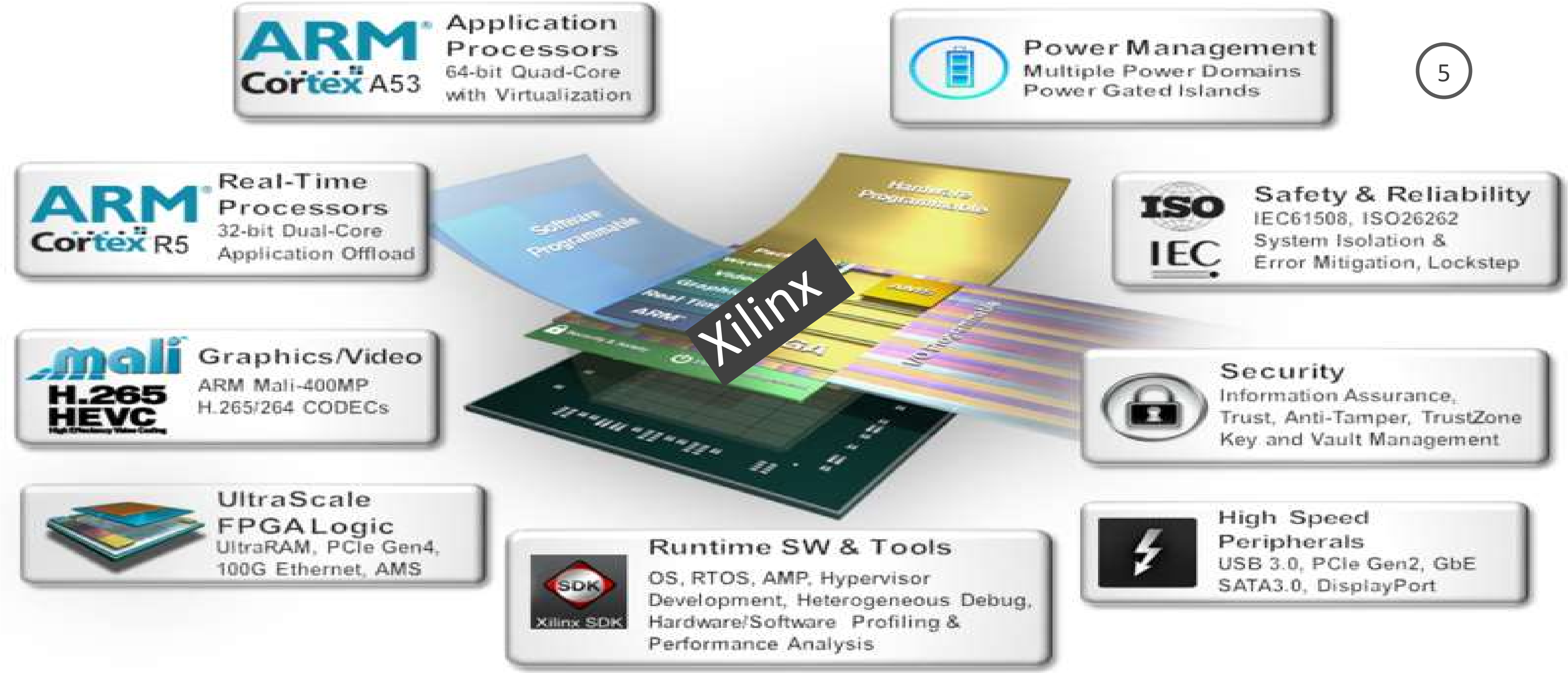
- Lots of data
- Simple computation/item
- Massive parallelism

CPU DSP, Accel GPU, ISP

Feedback loop
Noise removal
Noise removal
Segmentation & filtering
Pixel processing
Feature reduction
Image pyramids
Feature classification
Edge detection
Computation & processing
Object tracking
Gradient detection
Object detection
Avoidance signalling
Augmentation
Optical flow

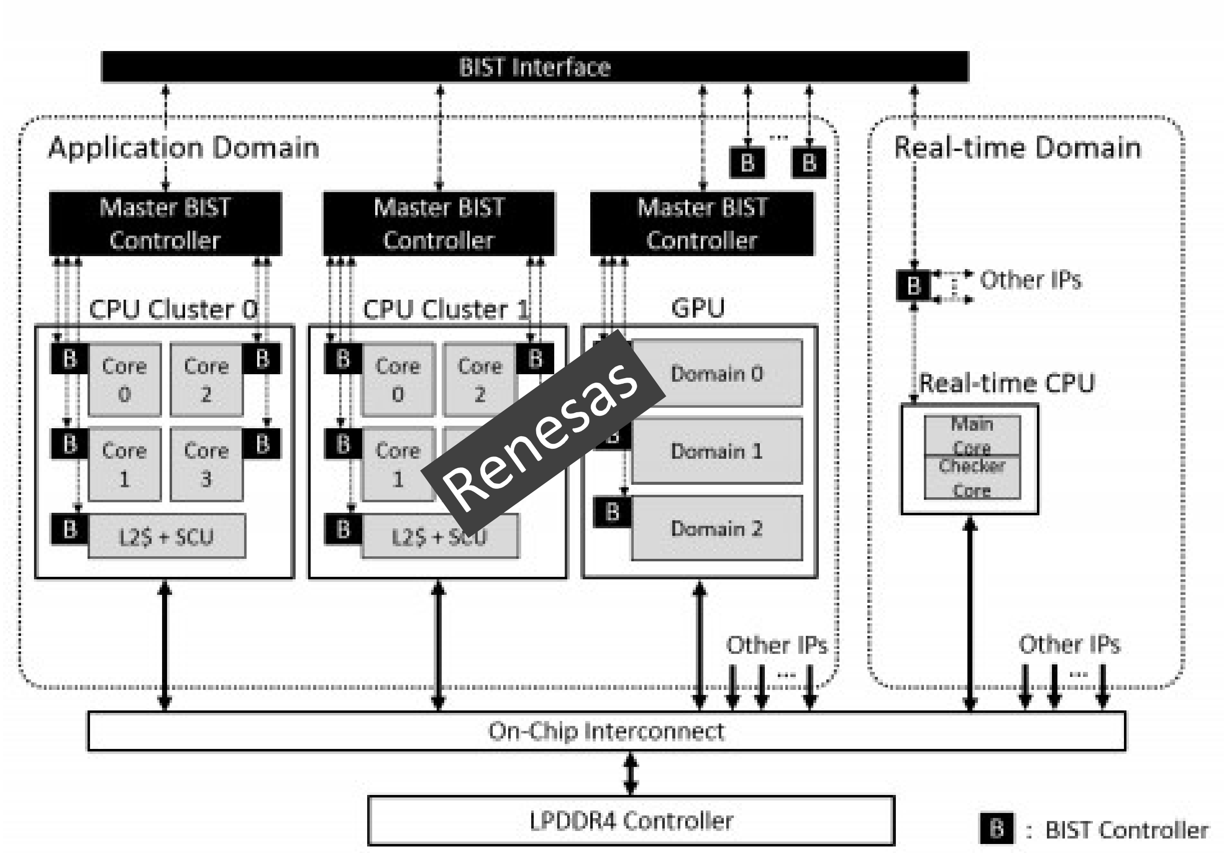
Pattern Recognition	Feature reduction Feature classification Augmentation
Feedback and Action	Computation & processing Feedback loop Avoidance signalling

Source: Extreme Tech, Google, ARM



Heterogenous MPSoCs with Real-time Processors

MOTIVATION

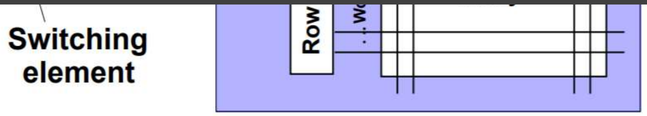


Heterogenous MPSoCs with Real-time Processors

MOTIVATION

- DRAM Consists of multiple banks

Bounding DRAM Interference in COTS Heterogeneous MPSoCs for Mixed Criticality Systems

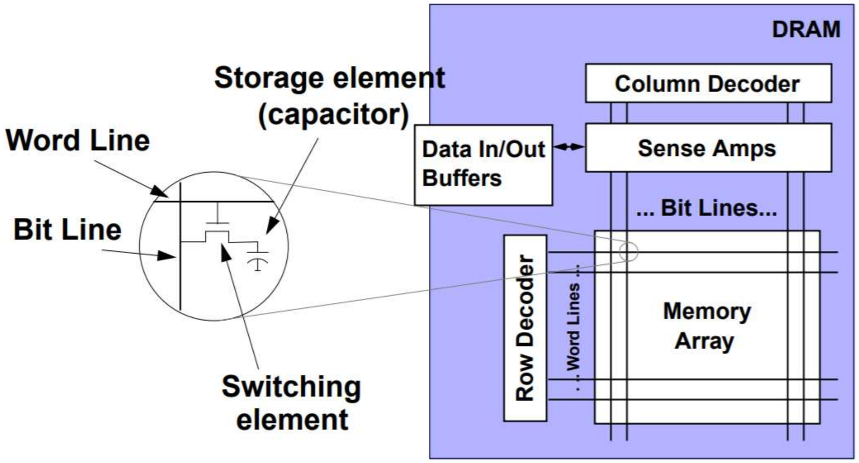


Background

DRAM

- DRAM Consists of multiple banks

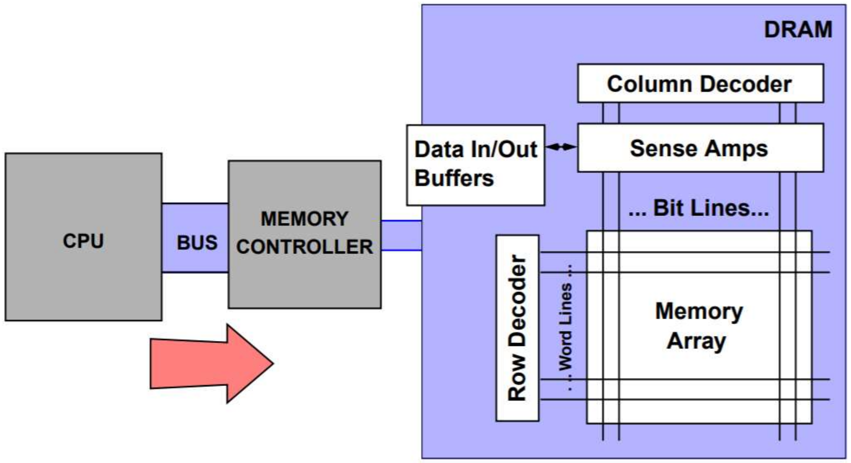
6



Background

DRAM

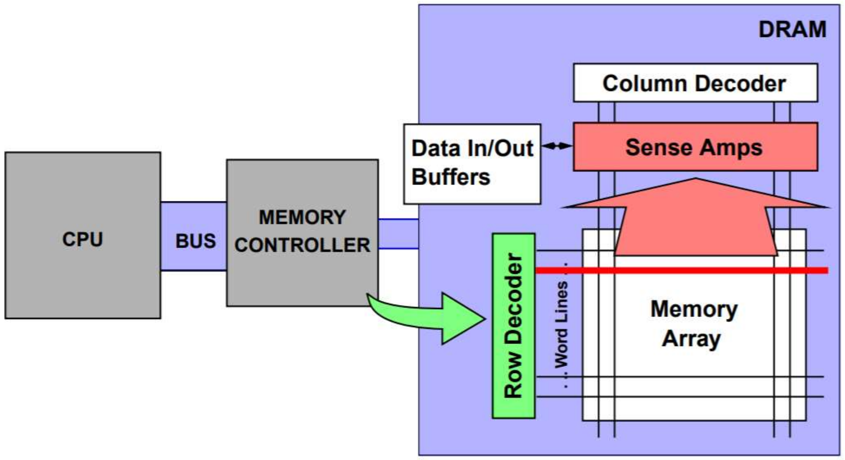
- DRAM Consists of multiple banks
- The memory controller (MC) manages accesses to DRAM



Background

DRAM

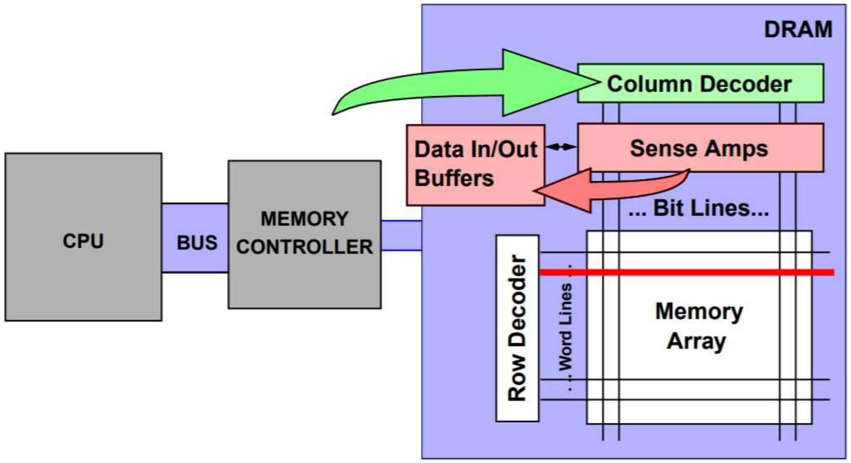
- DRAM Consists of multiple banks
- The memory controller (MC) manages accesses to DRAM
- A request in general consists of:
  - ACTIVATE command:
    - Bring data row from cells into sense amplifiers



Background

DRAM

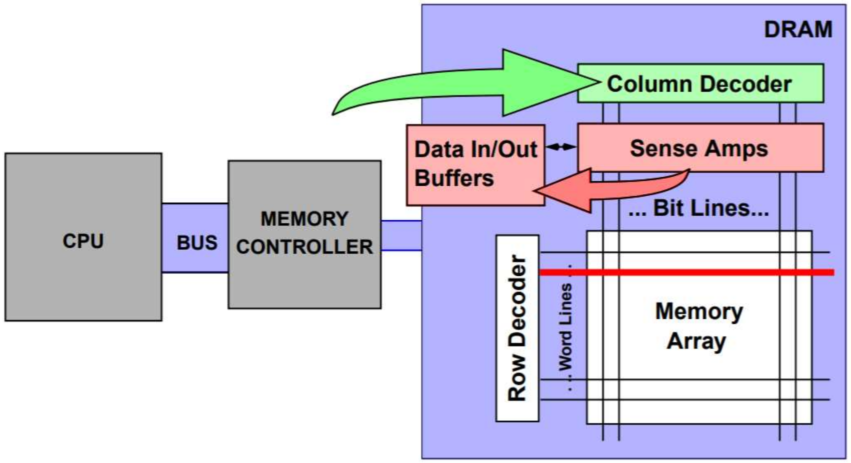
- DRAM Consists of multiple banks
- The memory controller (MC) manages accesses to DRAM
- A request in general consists of:
  - ACTIVATE command:
    - Bring data row from cells into sense amplifiers
  - RD/WR commands:
    - To read/write from specific columns in the sense amplifiers



Background

DRAM

- DRAM Consists of multiple banks
- The memory controller (MC) manages accesses to DRAM
- A request in general consists of:
  - ACTIVATE command:
    - Bring data row from cells into sense amplifiers
  - RD/WR commands:
    - To read/write from specific columns in the sense amplifiers
  - PRECHARGE command:
    - to write back a previous row in the sense amplifiers before bringing the new one



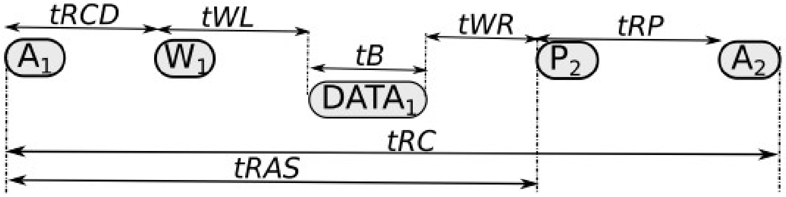
Background

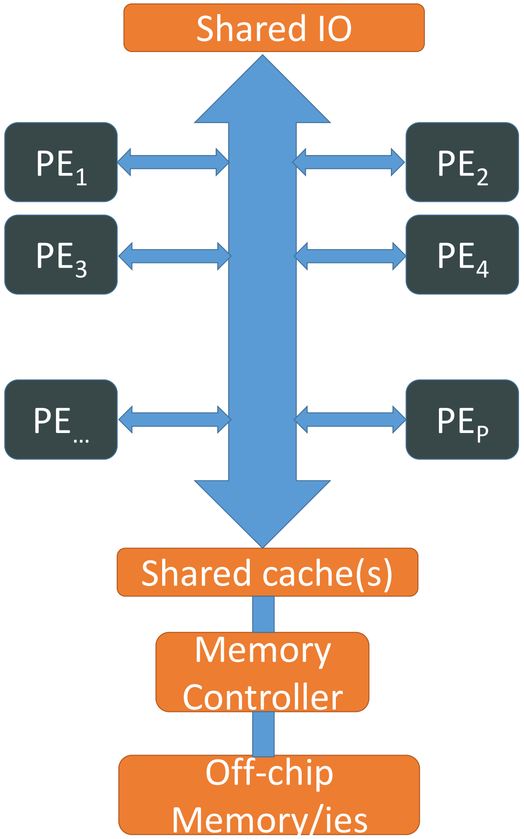
DRAM

Background

DRAM

- DRAM Consists of multiple banks
- The memory controller (MC) manages accesses to DRAM
- A request in general consists of:
  - ACTIVATE command:
    - Bring data row from cells into sense amplifiers
  - RD/WR commands:
    - To read/write from specific columns in the sense amplifiers
  - PRECHARGE command:
    - to write back a previous row in the sense amplifiers before bringing the new one
- All commands have associated timing constraints that have to be satisfied by the controller





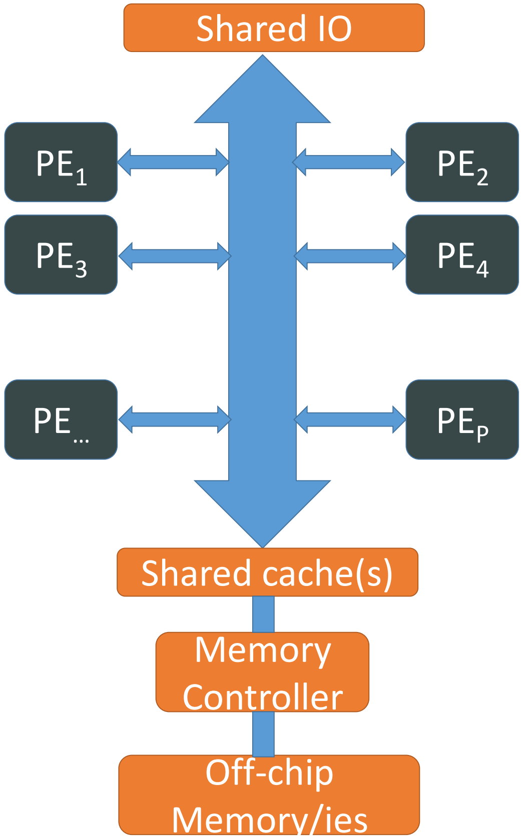
- P processing elements
  - P_{cr} critical + P_{ncr} non-critical
- LLC is write-back write-allocate
  - Writes to DRAM are only cache evictions
- Single-channel single-rank DRAM subsystem
- N_B DRAM banks

System Overview

MODEL

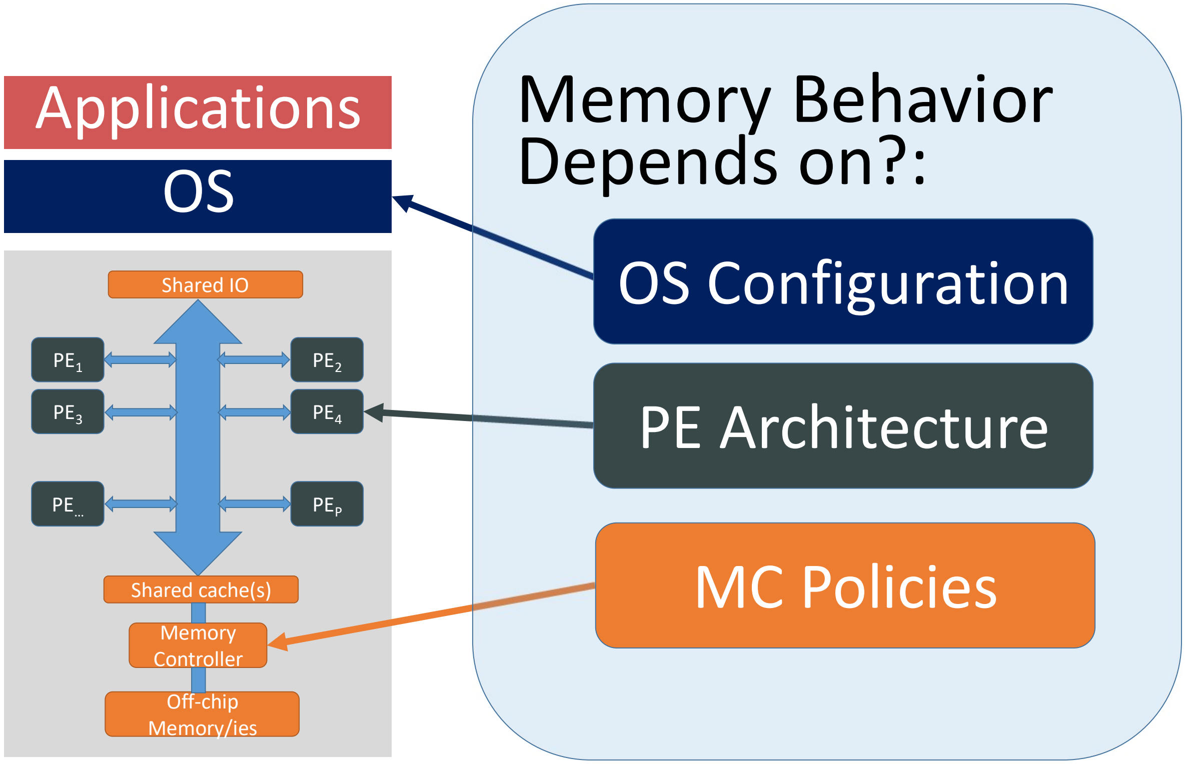
System Overview

MODEL



- P processing elements
  - P_{cr} critical + P_{ncr} non-critical
- LLC is write-back write-allocate
  - Writes to DRAM are only cache evictions
- Single-channel single-rank DRAM subsystem
- N_B DRAM banks

Goal:
Derive an upper bound on the delay incurred by any memory request of a critical PE



Memory Behavior Depends on?:

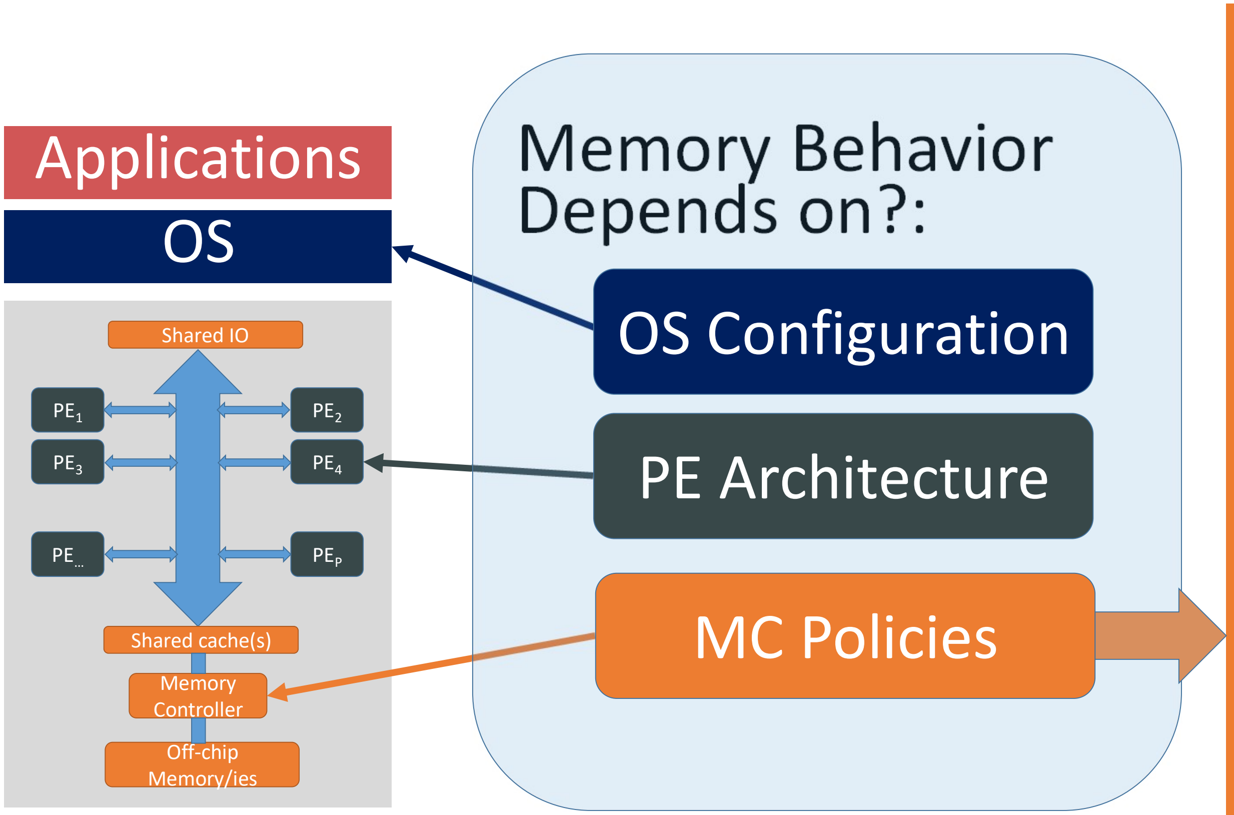
- OS Configuration
- PE Architecture
- MC Policies

System Details

MODEL

System Details

MODEL



Memory Behavior Depends on?:

OS Configuration

PE Architecture

MC Policies

- Priority:
  - PEs can be given priorities
  - COTS platforms support different priority levels
  - Existing analysis does not account for this
- Intra-bank scheduling
  - FR-FCFS
  - COTS also supports a threshold on reordering to prevent starvation
- Inter-bank scheduling
  - RR across banks
  - Two flavors:
    - Always schedule ready commands of any type (high performance)
    - Reorder only commands of different type (prevent starvation)
- Read/Write arbitration, two flavors:
  - Reads and writes have same priority
  - Serve in batches, where reads have higher priority

R/W Reorder

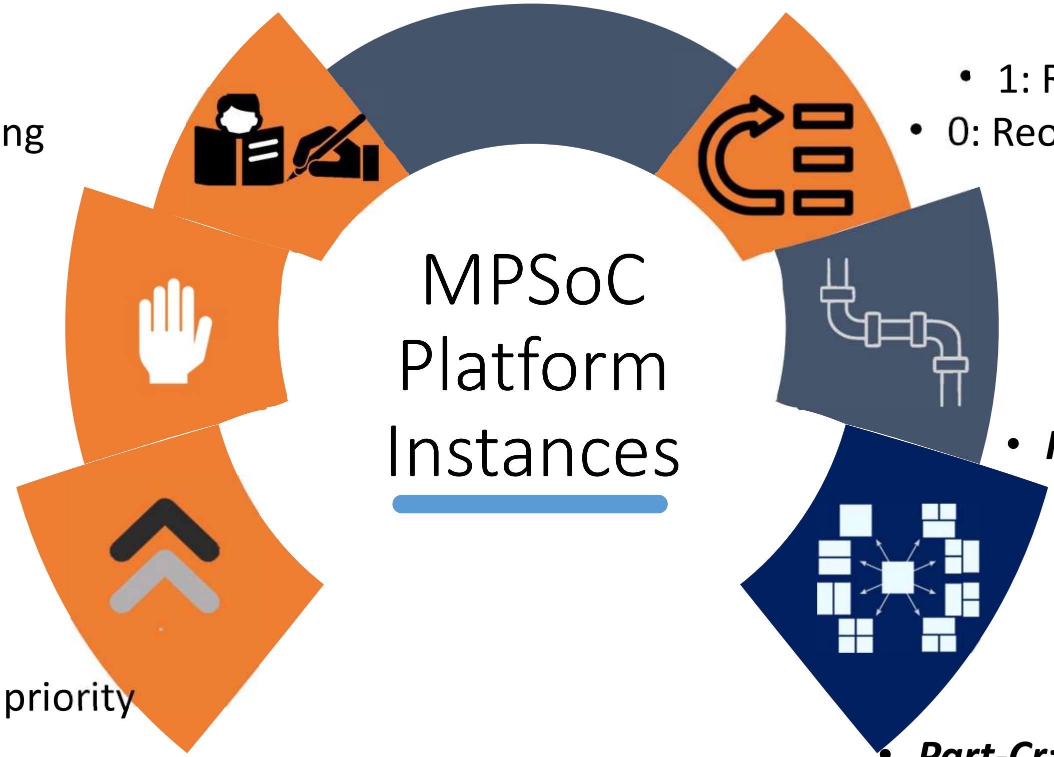
- 1: write batching
- 0: no write batching

FR-FCFS Threshold

- 1: FR-FCFS is capped
- 0: no cap on FR-FCFS

Priority

- 1: Critical PEs are higher priority
- 0: no priority



MPSoC Platform Instances

Inter-bank Reorder

- 1: Reorder across all commands
- 0: Reorder commands of diff types

Pipeline

- ***IO-All***: All PEs are In-order
- ***IO-Cr***: Critical PEs are in-order
- ***OOO-All***: All PEs are OOO

Partitioning

- ***No-Part***: No Partitioning
- ***Part-Cr***: Partition among critical apps
- ***Part-All***: Partition among all apps

Platform Instances

MODEL

Platform Instances

MODEL

R/W Reorder

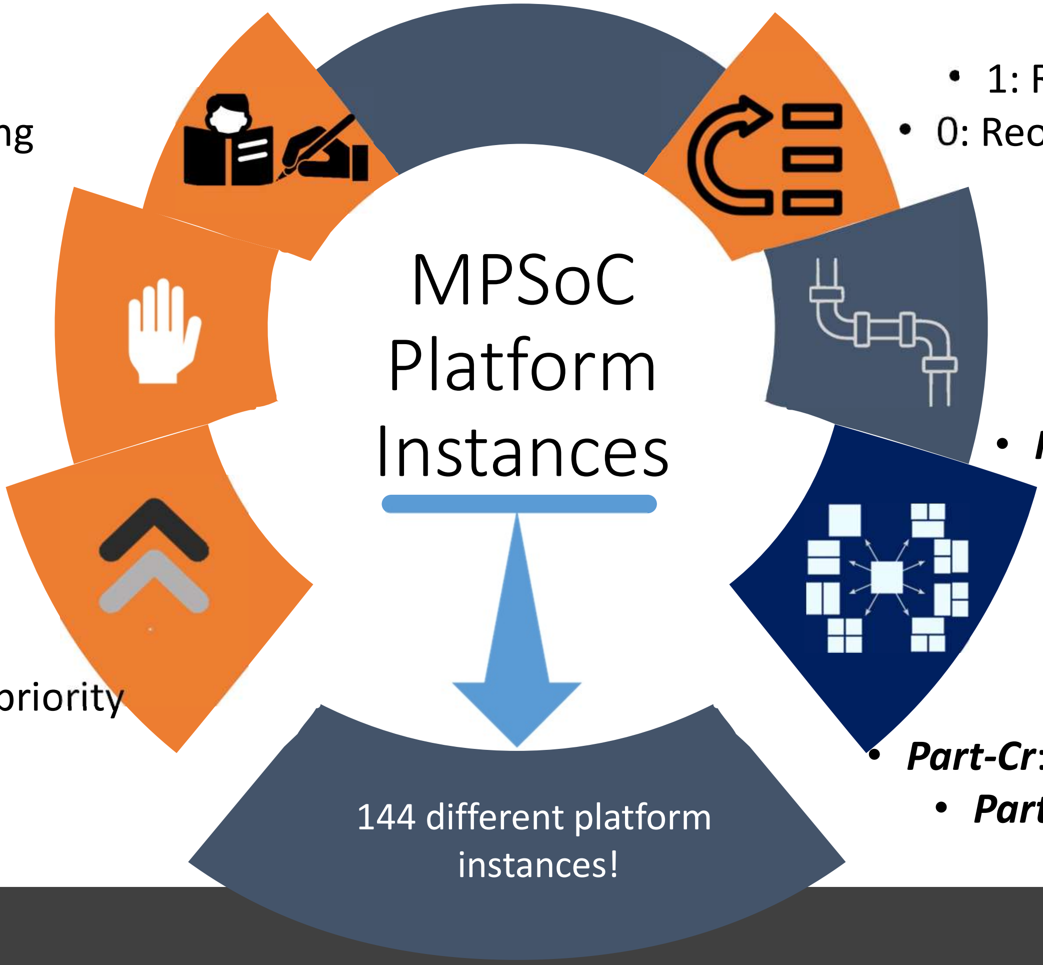
- 1: write batching
- 0: no write batching

FR-FCFS Threshold

- 1: FR-FCFS is capped
- 0: no cap on FR-FCFS

Priority

- 1: Critical PEs are higher priority
- 0: no priority



Inter-bank Reorder

- 1: Reorder across all commands
- 0: Reorder commands of diff types

Pipeline

- ***IO-All***: All PEs are In-order
- ***IO-Cr***: Critical PEs are in-order
- ***OOO-All***: All PEs are OOO

Partitioning

- ***No-Part***: No Partitioning
- ***Part-Cr***: Partition among critical apps
- ***Part-All***: Partition among all apps

144

OS	HW setup													
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0												
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

144 Platform Instances

General Observations

METHODOLOGY

144

OS	HW setup													
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0												
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

Observation 1:

Unboundedness of inter-bank RR with reordering

If RR reorders across all commands (breorder=1) and no write batching is deployed (wb=0) → unbounded WCD

General Observations

METHODOLOGY

General Observations

METHODOLOGY

108

OS	HW setup													
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0				UNBOUNDED								
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

108

10

OS			HW setup											
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0												
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

Observation 2:
Write batching effect

Write batching cancels the effect of RR breorder:
If wb=1 → breorder=x

General Observations

METHODOLOGY

72

OS	HW setup													
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0				UNBOUNDED						Same as wb=1,breorder=0		
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

General Observations

METHODOLOGY

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0				UNBOUNDED					
	0	1									
	1	0									
	1	1									
No-Part	0	0									
	0	1									
	1	0									
	1	1									
Part-Cr	0	0									
	0	1									
	1	0									
	1	1									

72

General Observations

METHODOLOGY

72

General Observations

METHODOLOGY

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0				🟥	🟥	🟥			
	0	1				🟥	🟥	🟥			
	1	0				🟥	🟥	🟥			
	1	1				🟥	🟥	🟥			
No-Part	0										
	0										
	1										
	1	1				🟥	🟥	🟥			
Part-Cr	0	0				🟥	🟥	🟥			
	0	1				🟥	🟥	🟥			
	1	0				🟥	🟥	🟥			
	1	1				🟥	🟥	🟥			

Observation 3:
Unboundedness of FR-FCFS without threshold
If thr=0 & ((No-Part) || ((Part-Cr) & pr=0)→ Unbounded WCD

10

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0				UNBOUNDED					
	0	1									
	1	0									
	1	1									
No-Part	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1									
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1									

54

General Observations

METHODOLOGY

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0									
	0	1									
	1	0									
	1										
No-Part	0										D
	0										
	1										
	1	1									
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1									

Observation 4:
Part-All effect
If Part-All → r_{ua} does not suffer Intra-bank reordering or conflict interferences:

- thr=x
- If wb=0 → pipe=x

General Observations

METHODOLOGY

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1			UNBOUNDED			confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1	1	confg2						confg14	confg15	confg16
No-Part	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1									
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1									

38

General Observations

METHODOLOGY

General Observations

METHODOLOGY

38

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1						confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1										6
No-Part	0										
	0										
	1										
	1										
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1									

Observation 5:

Part-Cr effect when wb=0

If Part-Cr & wb=0 → r_{ua} does not suffer Intra-bank reordering nor conflict interferences from critical PEs:

- IO-Cr and OOO-All have same effect on WCD

35

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1			UNBOUNDED			confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1	1	confg2						confg14	confg15	confg16
No-Part	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1									
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1	confg8								
	1	0	confg9								
	1	1	confg10								

General Observations

METHODOLOGY

35

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1						confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1										6
No-Part	0										
	0										
	1										
	1										
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1	confg8								
	1	0	confg9								
	1	1	confg10								

Observation 6:
Priority effect when wb=0

If pr=1 & wb=0 → pipeline architecture of non-critical PEs has no effect on WCD:

- IO-Cr and IO-All have same effect on WCD

General Observations

METHODOLOGY

10

<table>
<tr><th>OS</th><th colspan="11">HW setup</th></tr>
<tr><td rowspan="2">part</td><td rowspan="2">thr</td><td rowspan="2">pr</td><td colspan="3">wb=0,breorder=0</td><td colspan="3">wb=0,breorder=1</td><td colspan="3">wb=1,breorder=x</td></tr>
<tr><td>OOO</td><td>IO-Cr</td><td>IO-All</td><td>OOO</td><td>IO-Cr</td><td>IO-All</td><td>OOO</td><td>IO-Cr</td><td>IO-All</td></tr>
<tr><td rowspan="4">Part-All</td><td>0</td><td>0</td><td colspan="3">confg1</td><td colspan="3" rowspan="12">UNBOUNDED</td><td>confg11</td><td>confg12</td><td>confg13</td></tr>
<tr><td>0</td><td>1</td><td colspan="3">confg2</td><td>confg14</td><td>confg15</td><td>confg16</td></tr>
<tr><td>1</td><td>0</td><td colspan="3">confg1</td><td>confg11</td><td>confg12</td><td>confg13</td></tr>
<tr><td>1</td><td>1</td><td colspan="3">confg2</td><td>confg14</td><td>confg15</td><td>confg16</td></tr>
<tr><td rowspan="4">No-Part</td><td>0</td><td>0</td><td colspan="3" rowspan="2">UNBOUNDED</td><td colspan="3" rowspan="2">UNBOUNDED</td></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>1</td><td></td><td colspan="2">Confg7</td><td></td><td></td><td></td></tr>
<tr><td rowspan="4">Part-Cr</td><td>0</td><td>0</td><td colspan="3">UNBOUNDED</td><td colspan="3">UNBOUNDED</td></tr>
<tr><td>0</td><td>1</td><td colspan="2">confg8</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>0</td><td colspan="2">confg9</td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>1</td><td colspan="2">confg10</td><td></td><td></td><td></td><td></td></tr>
</table>

34

General Observations

METHODOLOGY

10

34

General Observations

METHODOLOGY

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1						confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1										6
No-Part	0										
	0										
	1										
	1										
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1	confg8								
	1	0	confg9								
	1	1	confg10								

Observation 7:

Priority with Part-Cr effect

- thr=x
- If wb=0 → pipe=x

Same as Part-All effect!!

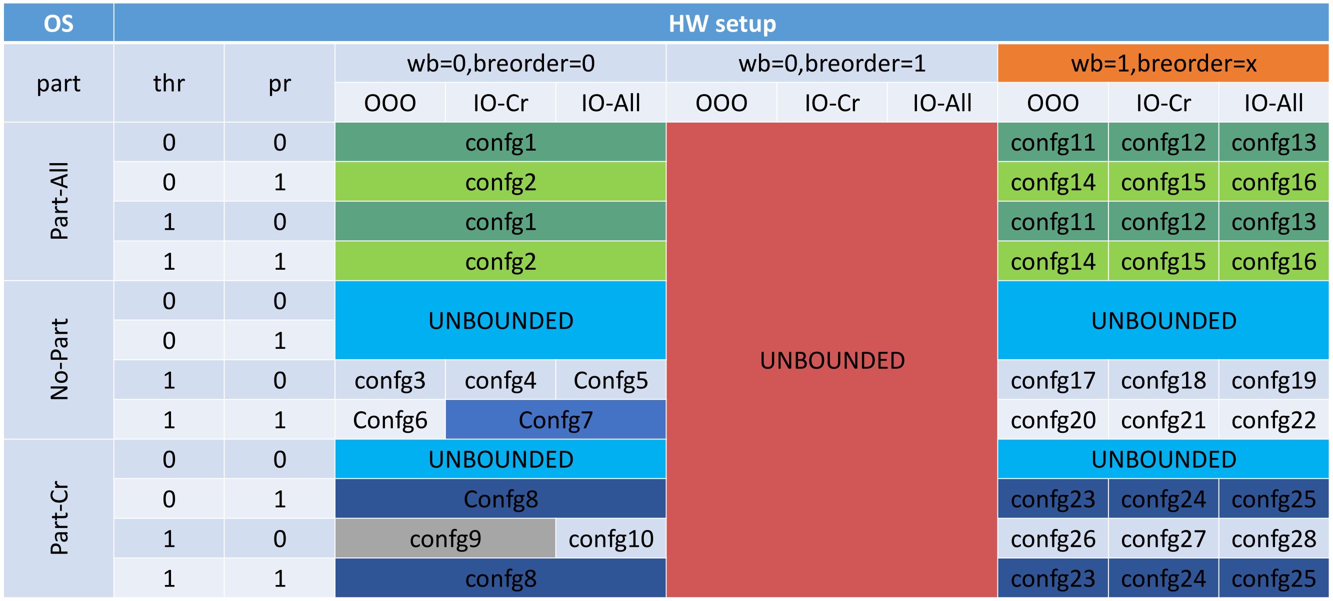
28

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1			UNBOUNDED			confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1	1	confg2						confg14	confg15	confg16
No-Part	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1		Confg7							
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1	confg8						confg23	confg24	confg25
	1	0	confg9								
	1	1	confg8						confg23	confg24	confg25

General Observations

METHODOLOGY

28



OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1			UNBOUNDED			confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1	1	confg2						confg14	confg15	confg16
No-Part	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0	confg3	confg4	Confg5				confg17	confg18	confg19
	1	1	Confg6	Confg7					confg20	confg21	confg22
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1	Confg8						confg23	confg24	confg25
	1	0	confg9		confg10				confg26	confg27	confg28
	1	1	confg8						confg23	confg24	confg25

144 Instances → 28 Configurations

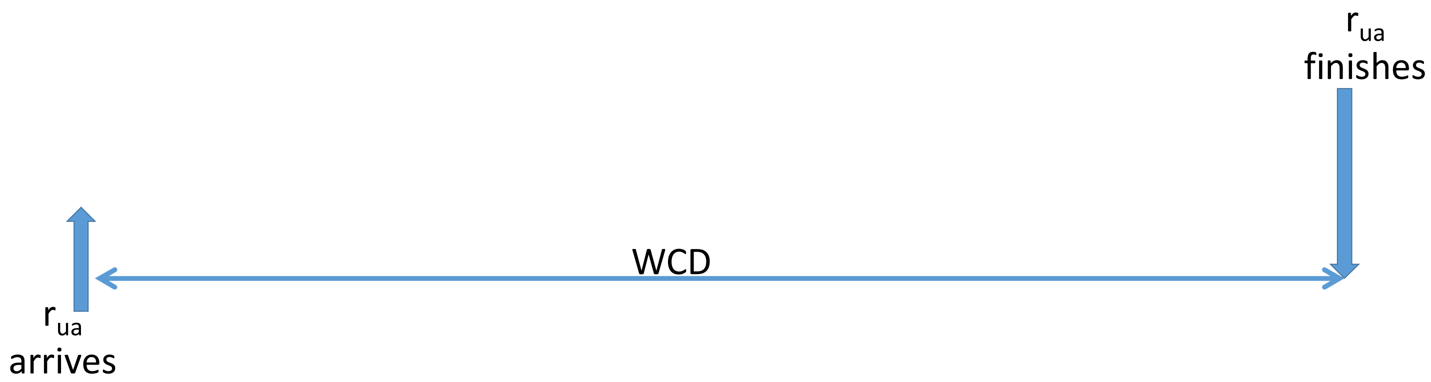
General Observations

METHODOLOGY

- Consider all timing constraints generated by commands of interfering requests of other PEs serviced between the times when r_{ua} arrives and finishes

- + Delays due to command bus contention

- Compute WCD for each configuration?

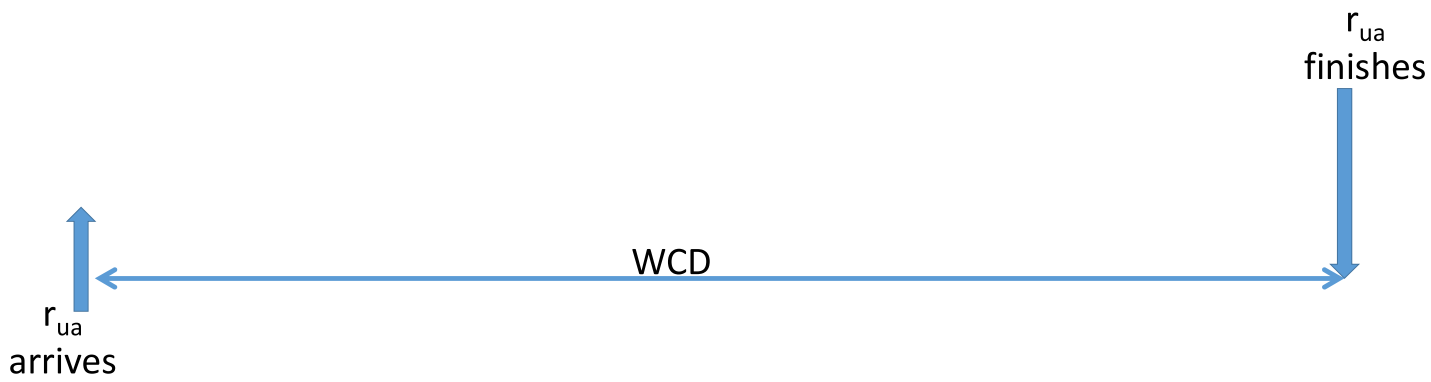


Memory Delay Building Blocks

Memory Delay Building Blocks

METHODOLOGY

- Consider all timing constraints generated by commands of interfering requests of other PEs serviced between the times when r_{ua} arrives and finishes
- + Delays due to command bus contention
- Compute WCD for each configuration? ☹
  - Still too much
  - Not general enough



Memory Delay Building Blocks

METHODOLOGY

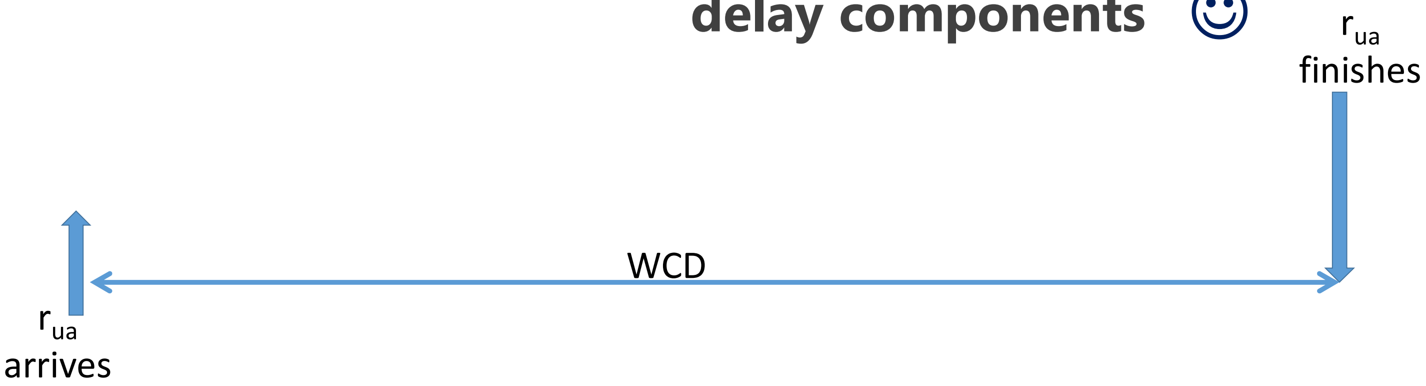
- Consider all timing constraints generated by commands of interfering requests of other PEs serviced between the times when r_{ua} arrives and finishes

- + Delays due to command bus contention

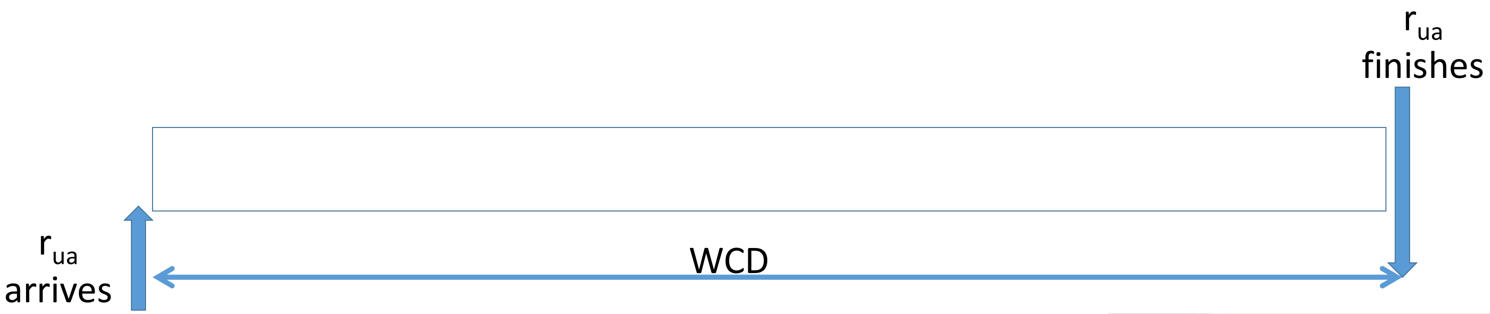
- Compute WCD for each configuration? ☹
  - Still too much
  - Not general enough

→

- **Configuration-independent DRAM delay components** ☺



- **We classify interfering requests (aka delay sources) into four types → <u>causing four basic interferences:</u>**

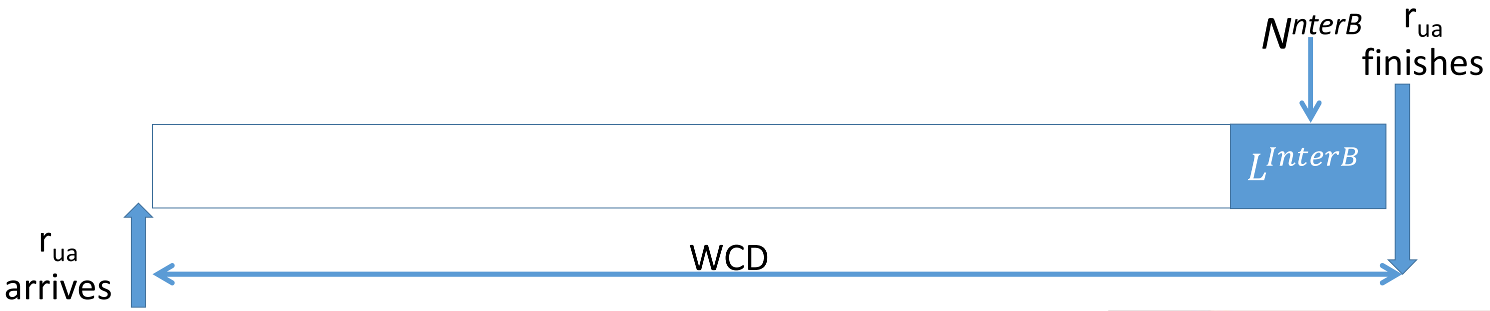


Memory Delay Building Blocks

METHODOLOGY

- **We classify interfering requests (aka delay sources) into four types → <u>causing four basic interferences:</u>**
  1. Inter-bank interference (requests to other banks)

$$WCD = L^{InterB}(N^{InterB}, wb)$$



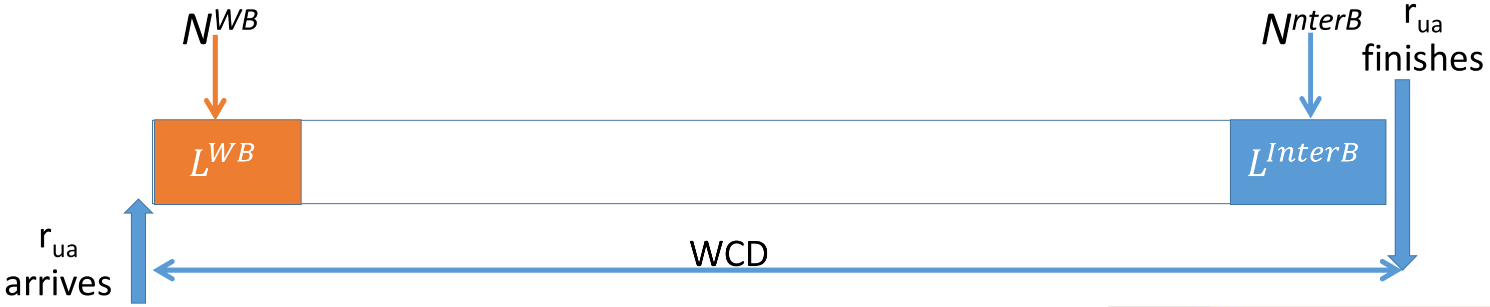
Memory Delay Building Blocks

Memory Delay Building Blocks

METHODOLOGY

- **We classify interfering requests (aka delay sources) into four types → <u>causing four basic interferences:</u>**
  1. Inter-bank interference (requests to other banks)
  2. Write batch Interference (only for R/W reordering)

$$WCD = L^{InterB}(N^{InterB}, wb) + wb \times L^{WB}(N^{WB})$$

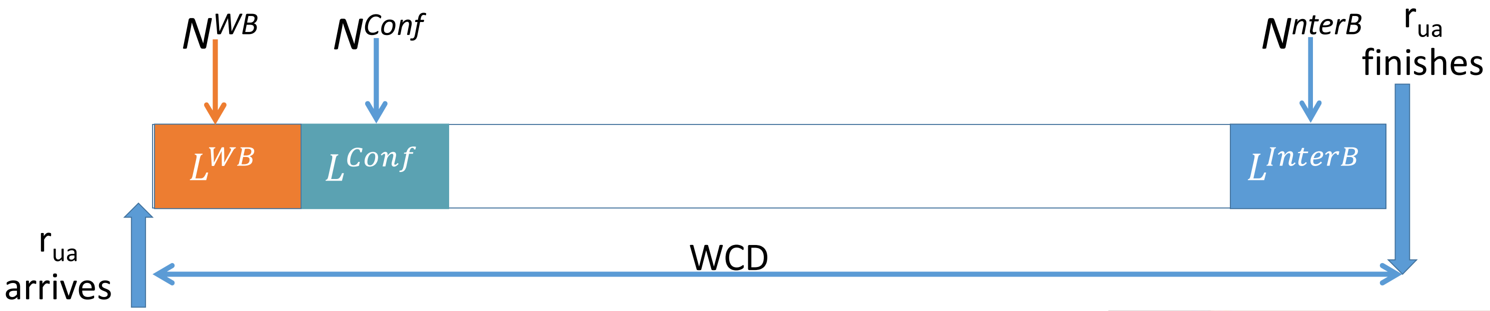


Memory Delay Building Blocks

METHODOLOGY

- **We classify interfering requests (aka delay sources) into four types → <u>causing four basic interferences:</u>**
  1. Inter-bank interference (requests to other banks)
  2. Write batch Interference (only for R/W reordering)
  3. Conflict interference (requests to same bank different rows arrived before r_{ua})

$$WCD = L^{InterB}(N^{InterB}, wb) + wb \times L^{WB}(N^{WB}) + L^{Conf}(N^{Conf})$$

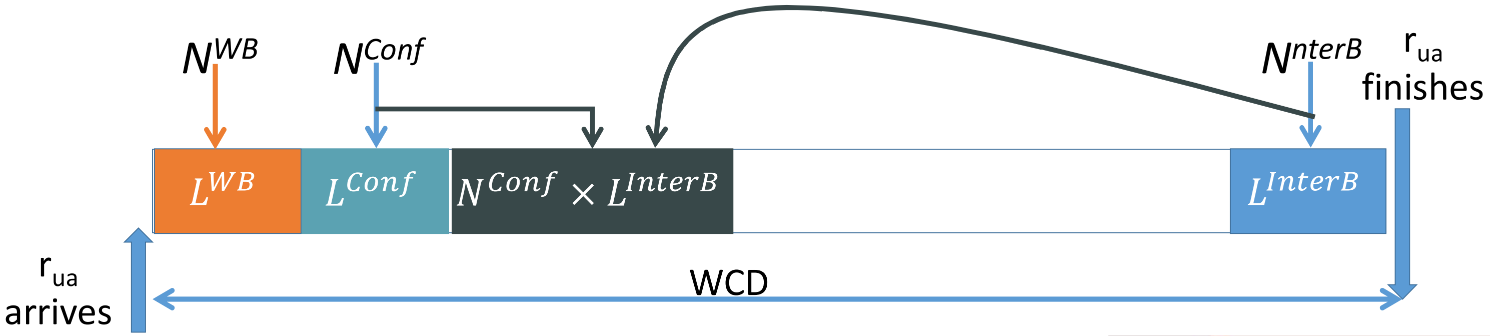


Memory Delay Building Blocks

METHODOLOGY

- **We classify interfering requests (aka delay sources) into four types → <u>causing four basic interferences:</u>**
  1. Inter-bank interference (requests to other banks)
  2. Write batch Interference (only for R/W reordering)
  3. Conflict interference (requests to same bank different rows arrived before r_{ua})

$$
\begin{gathered}
WCD = \\
L^{InterB}(N^{InterB}, wb) \\
+wb \times L^{WB}(N^{WB}) \\
+L^{Conf}(N^{Conf}) \\
+N^{Conf} \times L^{InterB}(N^{InterB}, wb)
\end{gathered}
$$

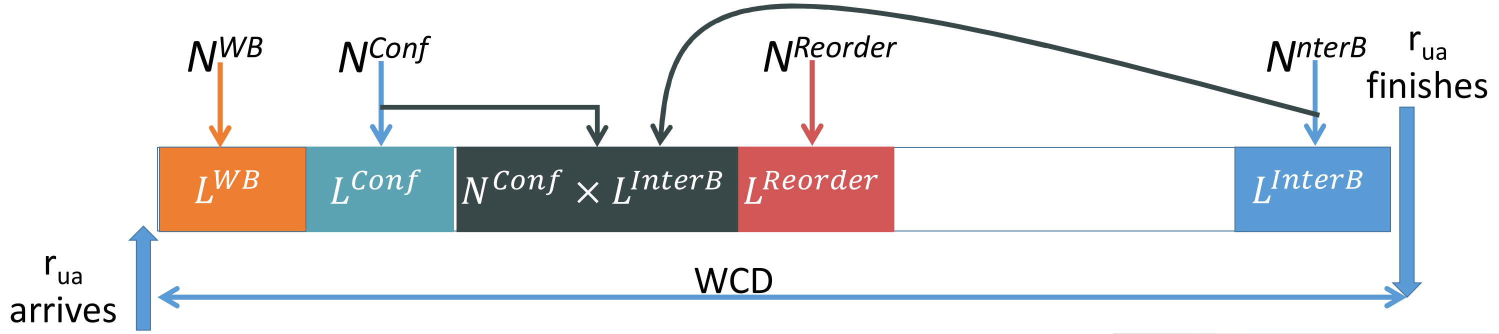


Memory Delay Building Blocks

METHODOLOGY

- **We classify interfering requests (aka delay sources) into four types → <u>causing four basic interferences:</u>**
  1. Inter-bank interference (requests to other banks)
  2. Write batch Interference (only for R/W reordering)
  3. Conflict interference (requests to same bank different rows arrived before r_{ua})
  4. Intra-bank Reorder interference (FR-FCFS)

$$
\begin{gathered}
WCD = \\
L^{InterB}(N^{InterB}, wb) \\
+ wb \times L^{WB}(N^{WB}) \\
+ L^{Conf}\left(N^{Conf}\right) \\
+ N^{Conf} \times L^{InterB}(N^{InterB}, wb) \\
+ L^{Reorder}\left(N^{Reorder}, wb\right)
\end{gathered}
$$

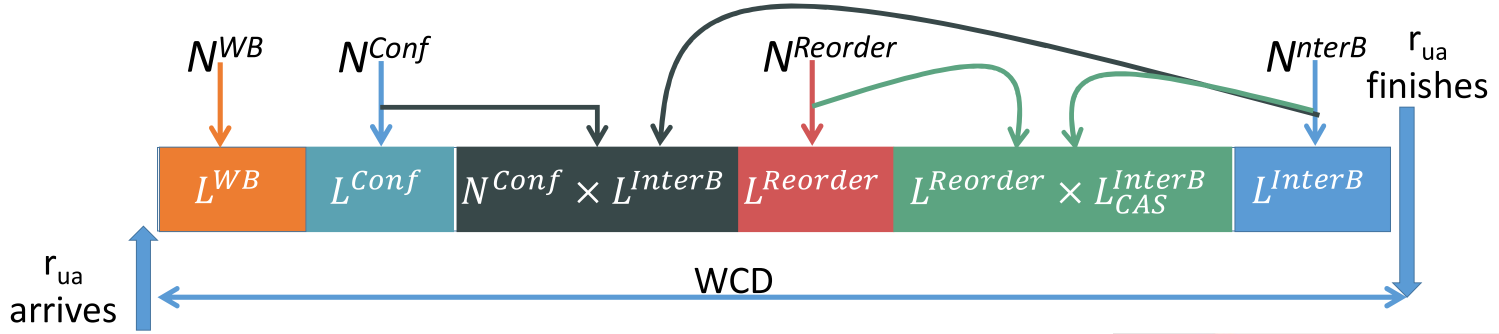


Memory Delay Building Blocks

METHODOLOGY

- **We classify interfering requests (aka delay sources) into four types → <u>causing four basic interferences:</u>**
  1. Inter-bank interference (requests to other banks)
  2. Write batch Interference (only for R/W reordering)
  3. Conflict interference (requests to same bank different rows arrived before r$_{ua}$)
  4. Intra-bank Reorder interference (FR-FCFS)

$$
\begin{gathered}
WCD = \\
L^{InterB}(N^{InterB}, wb) \\
+wb \times L^{WB}(N^{WB}) \\
+L^{Conf}\left(N^{Conf}\right) \\
+N^{Conf} \times L^{InterB}(N^{InterB}, wb) \\
+\ L^{Reorder}\left(N^{Reorder}, wb\right) \\
+N^{Reorder} \times L^{InterB}_{CAS}(N^{InterB}, wb)
\end{gathered}
$$



Memory Delay Building Blocks

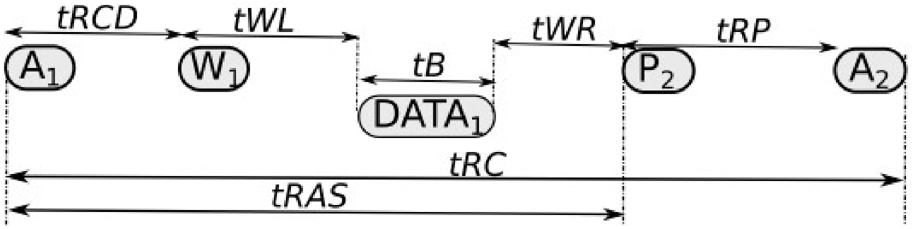
METHODOLOGY

- **Let's assume we know # of interfering requests (*Ns*), how to compute the latency components (*Ls*)?**
  - → *Ls* only depend on *Ns* and JEDEC "known" timing constraints

$$
\begin{gathered}
WCD = \\
L^{InterB}(N^{InterB}, wb) \\
+wb \times L^{WB}(N^{WB}) \\
+L^{Conf}\left(N^{Conf}\right) \\
+N^{Conf} \times L^{InterB}(N^{InterB}, wb) \\
+\ L^{Reorder}\left(N^{Reorder}, wb\right) \\
+N^{Reorder} \times L^{InterB}_{CAS}(N^{InterB}, wb)
\end{gathered}
$$

- **Let's assume we know # of interfering requests (*Ns*), how to compute the latency components (*Ls*)?**
  → *Ls* only depend on Ns and JEDEC "known" timing constraints
  → L^{Conf} as example

$$L^{Conf}(N^{Conf}) = N^{Conf} \times (MAX(tRAS, tRCD + tWL + tB + tWR) + tRP)$$



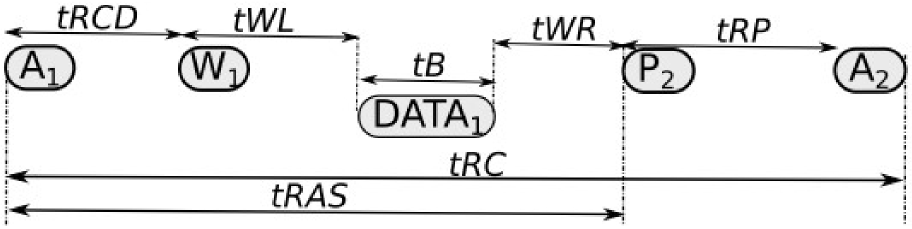
Memory Delay Building Blocks

METHODOLOGY

- **Let’s assume we know # of interfering requests (*Ns*), how to compute the latency components (*Ls*)?**
  → *Ls* only depend on Ns and JEDEC “known” timing constraints
  → L^{Conf} as example

- **Configuration-independent DRAM delay components** ☺

$$L^{Conf}(N^{Conf}) = N^{Conf} \times (MAX(tRAS, tRCD + tWL + tB + tWR) + tRP)$$



Memory Delay Building Blocks

Memory Delay Building Blocks

METHODOLOGY

- **Let’s assume we know # of interfering requests (*Ns*), how to compute the latency components (*Ls*)?**

  → *Ls* only depend on Ns and JEDEC “known” timing constraints

- **Configuration-independent DRAM delay components** ☺

- **Now: It only remains to compute the *Ns*.**

- Now: It only remains to compute the *Ns*. → Config. dependent

# of Interfering Requests

# of Interfering Requests

METHODOLOGY

- Now: It only remains to compute the *Ns*. → Config. dependent
- Take confg3 as an example:

confg3:

- no WB
- FR-FCFS thr
- no FP
- Inter-bank reorder among different types only (breorder=0)
- All PEs are OOO
- no partitioning

- Now: It only remains to compute the *Ns*. → Config. dependent
- Take confg3 as an example:

1. **Conflicts ($\boldsymbol{N^{Conf}}$):**
   - OOO-All → each PE has PR pending reqs
   - No FP → critical and non-critical scheduled similarly
   - Then $\boldsymbol{N^{Conf}} = (\boldsymbol{P} - \mathbf{1}) \times \boldsymbol{PR}$ requests can conflict with r_{ua}

confg3:

- no WB
- FR-FCFS thr
- no FP
- Inter-bank reorder among different types only (breorder=0)
- All PEs are OOO
- no partitioning

# of Interfering Requests

METHODOLOGY

# of Interfering Requests

METHODOLOGY

- Now: It only remains to compute the *Ns*. → Config. dependent
- Take confg3 as an example:

1. **Conflicts (N^{Conf}):**
   - OOO-All → each PE has PR pending reqs
   - No FP → critical and non-critical scheduled similarly
   - Then $N^{Conf} = (P - 1) \times PR$ requests can conflict with r_{ua}
2. **Reorder ($N^{Reorder}$):**
   - FR-FCFS thr → max of $N^{Reorder} = N_{thr}$ requests can be reordered before r_{ua}

confg3:

- no WB
- FR-FCFS thr
- no FP
- Inter-bank reorder among different types only (breorder=0)
- All PEs are OOO
- no partitioning

# of Interfering Requests

- Now: It only remains to compute the *Ns*. → Config. dependent
- Take confg3 as an example:

1. **Conflicts (N^{Conf}):**
   - OOO-All → each PE has PR pending reqs
   - No FP → critical and non-critical scheduled similarly
   - Then $N^{Conf} = (P - 1) \times PR$ requests can conflict with r_{ua}
2. **Reorder ($N^{Reorder}$):**
   - FR-FCFS thr → max of $N^{Reorder} = N_{thr}$ requests can be reordered before r_{ua}
3. **Inter-bank (N^{InterB}):**
   - RR arbiter and no FP→ max of $N^{InterB} = N_B - 1$ reqs from other banks can be reordered before r_{ua}

confg3:

- no WB
- FR-FCFS thr
- no FP
- Inter-bank reorder among different types only (breorder=0)
- All PEs are OOO
- no partitioning

- Now: It only remains to compute the *Ns*. → Config. dependent
- Take confg3 as an example:

1. **Conflicts (N^{Conf}):**
   - OOO-All → each PE has PR pending reqs
   - No FP → critical and non-critical scheduled similarly
   - Then $N^{Conf} = (P-1) \times PR$ requests can conflict with r_{ua}
2. **Reorder ($N^{Reorder}$):**
   - FR-FCFS thr → max of $N^{Reorder} = N_{thr}$ requests can be reordered before r_{ua}
3. **Inter-bank (N^{InterB}):**
   - RR arbiter and no FP→ max of $N^{InterB} = N_B - 1$ reqs from other banks can be reordered before r_{ua}
4. **Write Batch (N^{WB})**
   - No WB → N^{WB}=0

confg3:

- no WB
- FR-FCFS thr
- no FP
- Inter-bank reorder among different types only (breorder=0)
- All PEs are OOO
- no partitioning

# of Interfering Requests

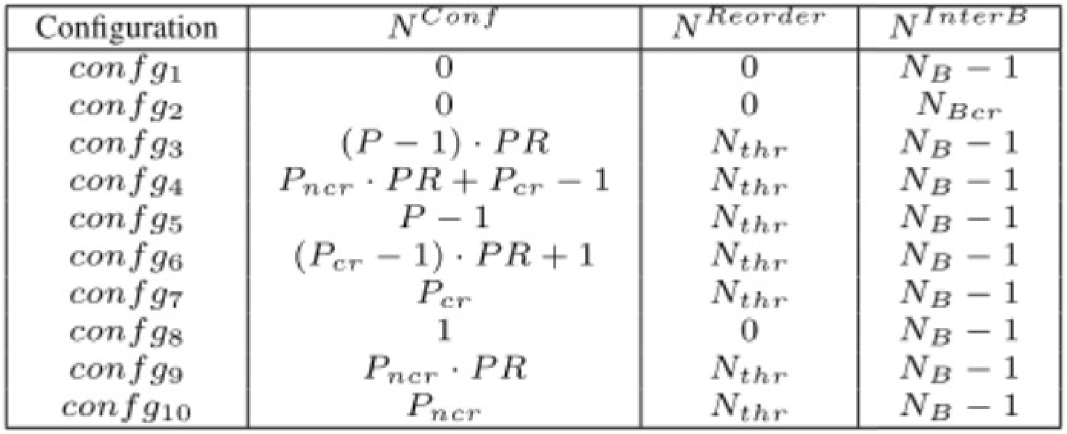
# of Interfering Requests

METHODOLOGY

- Now: It only remains to compute the *Ns*. → Config. dependent
- Take config3 as an example.

- **Follow Same approach for all configurations**

1. **Conflicts (N^{Conf}):**
   - OOO-All → each PE has PR pending reqs
   - No FP → critic
   - Then N^{Conf} =
2. **Reorder (N^{Reorde}**
   - FR-FCFS thr → reordered bef
3. **Inter-bank (N^{Inte}**
   - RR arbiter and other banks can be reordered before r_{ua}
4. **Write Batch (N^{WB})**
   - No WB → N^{WB}=0



Configuration	N^{Conf}	$N^{Reorder}$	N^{InterB}
$config_1$	0	0	N_B-1
$config_2$	0	0	N_{Bcr}
$config_3$	$(P-1)\cdot PR$	N_{thr}	N_B-1
$config_4$	$P_{ncr}\cdot PR+P_{cr}-1$	N_{thr}	N_B-1
$config_5$	$P-1$	N_{thr}	N_B-1
$config_6$	$(P_{cr}-1)\cdot PR+1$	N_{thr}	N_B-1
$config_7$	P_{cr}	N_{thr}	N_B-1
$config_8$	1	0	N_B-1
$config_9$	$P_{ncr}\cdot PR$	N_{thr}	N_B-1
$config_{10}$	P_{ncr}	N_{thr}	N_B-1

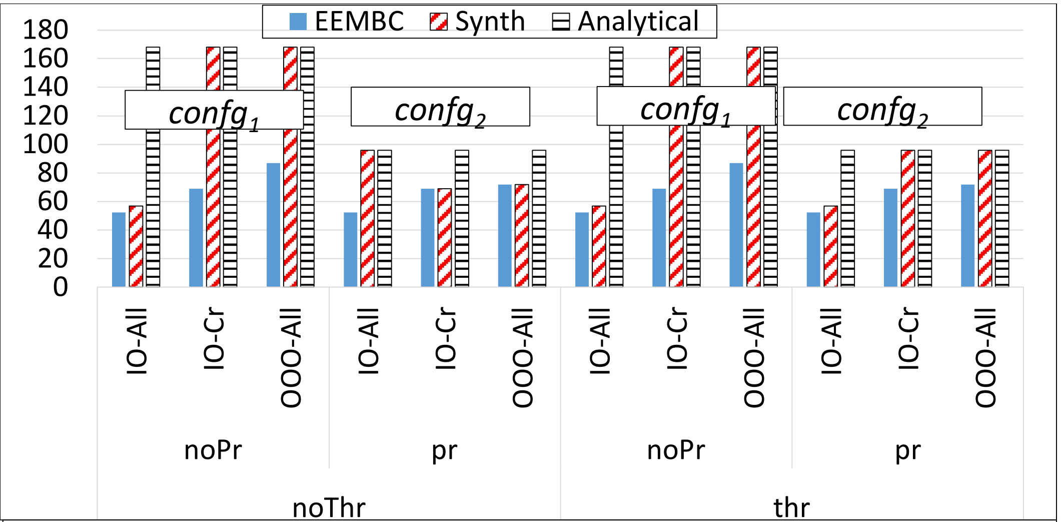
confg3:

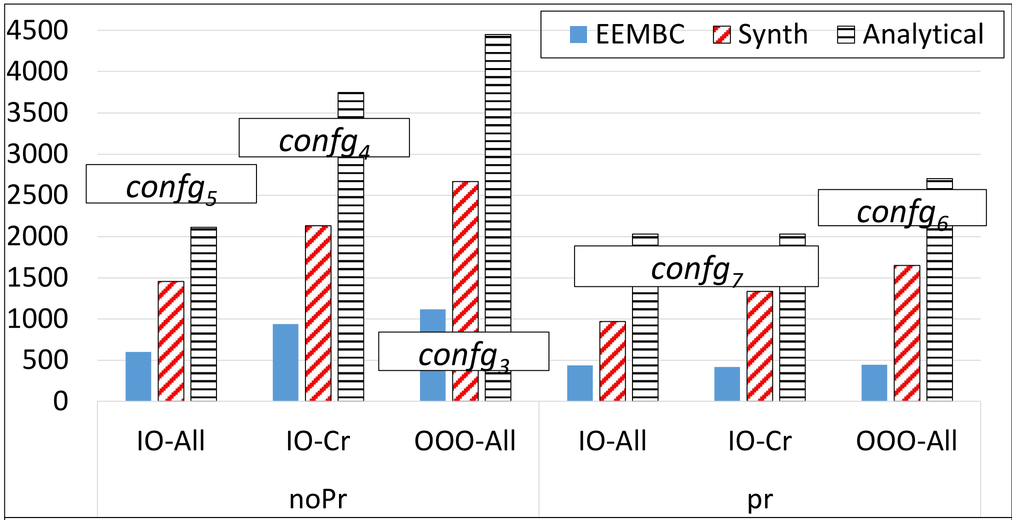
- no WB
- FR-FCFS thr
- FP
- r-bank reorder among rent types only order=0)
- PEs are OOO
- no partitioning

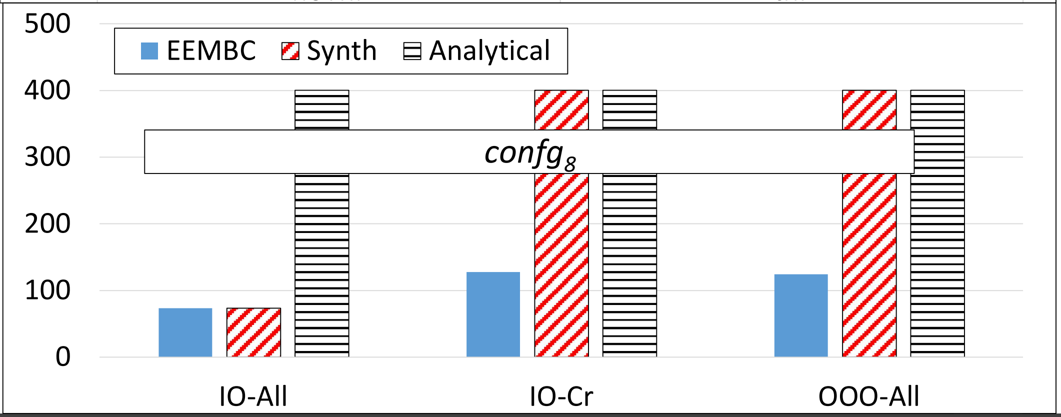
Evaluation Setup

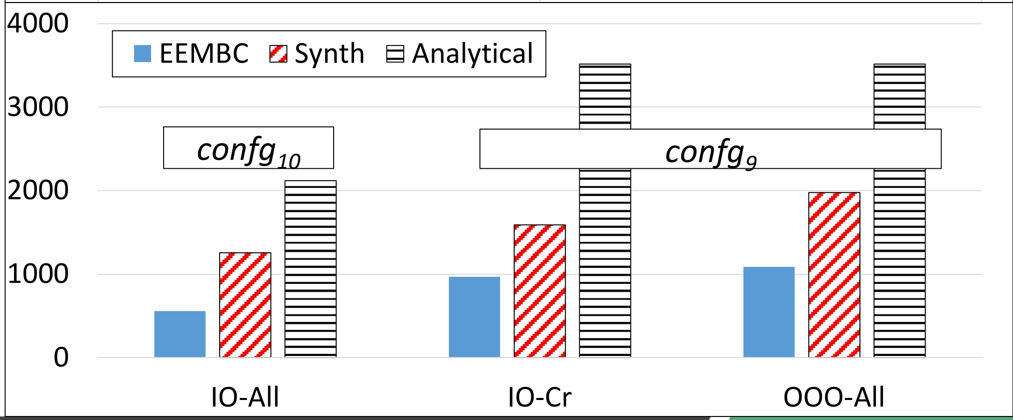
RESULTS

PEs	• A private 16KB L1 and a shared 1MB L2 cache • An in-order PE has a maximum of one pending request to the DRAM • An OOO PE has a maximum of 4 pending requests to the DRAM (PR = 4) • Four-processor system unless otherwise specified	
OS Mapping	• Through the virtual-to-physical address mapping component at MacSim’s frontend • Based on the configuration, we enable the corresponding partitioning (Part-All, Part-Cr, or No-Part)	
DRAM	DDR3-1333H with single channel, single rank, and 8 banks	
MC	• Based on the configuration, • Per-bank queues with RR among banks and FR-FCFS arbitration within each bank • Based on the configuration: • critical PEs can be assigned higher priority than non-critical PEs • enable or disable the threshold for FR-FCFS • For enabled threshold:$N_{thr} = 8$, unless otherwise specified • enable or disable write batching	
Benchmarks	EEMBC Automotive	• The two critical PEs execute a2time and rspeed • The two non-critical PEs execute matrix and aifftr
	Synthetic	• Each of the critical PEs execute one instance of the latency benchmark Each of the non-critical PEs execute one instance of the Bandwidth benchmark



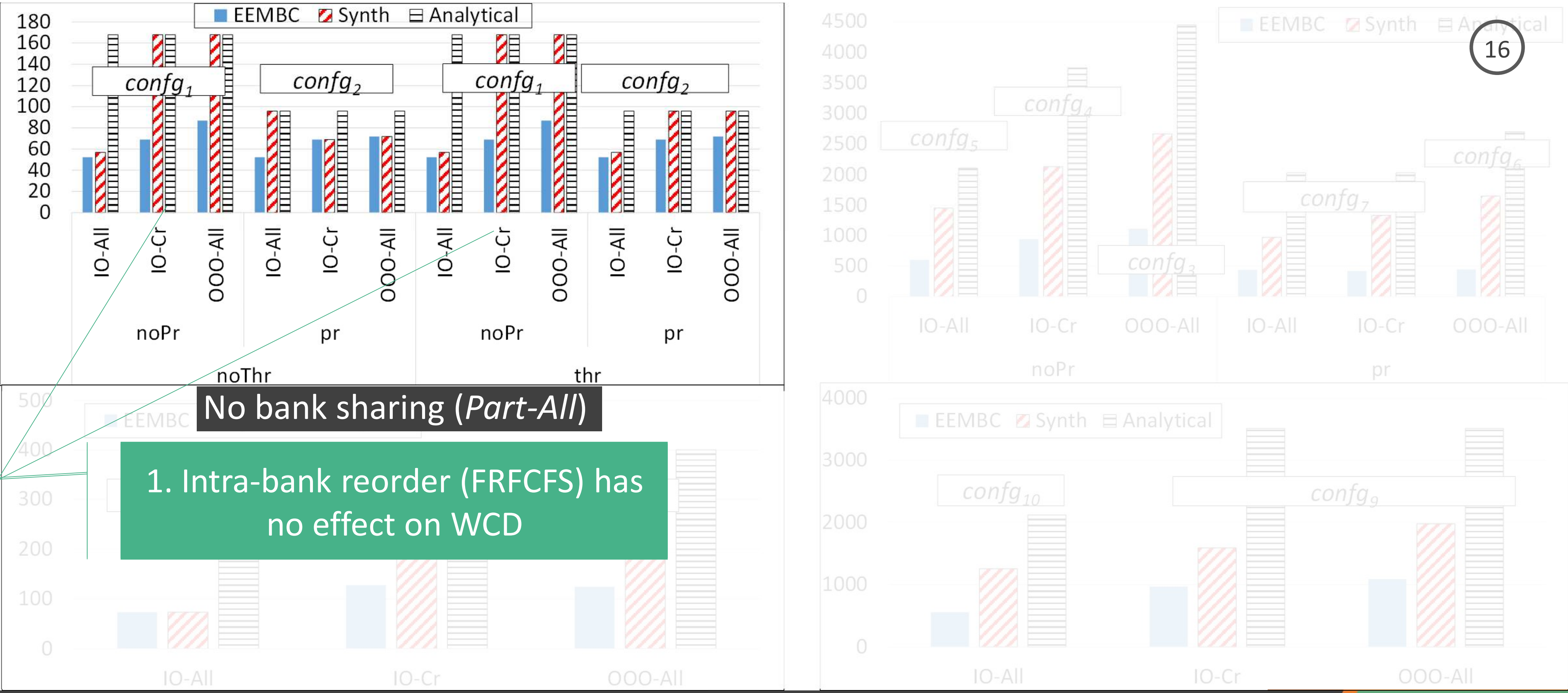






WCD of Critical Processors

RESULTS

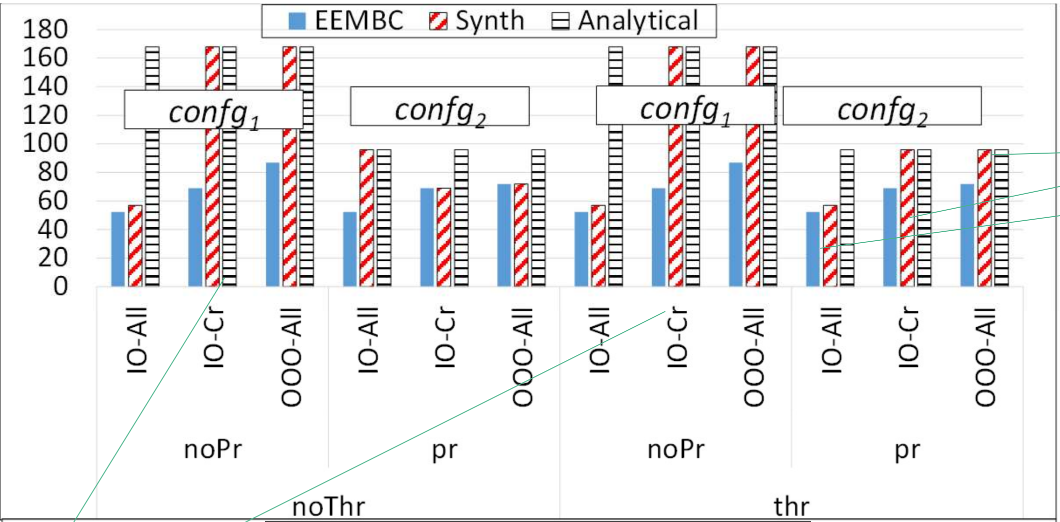


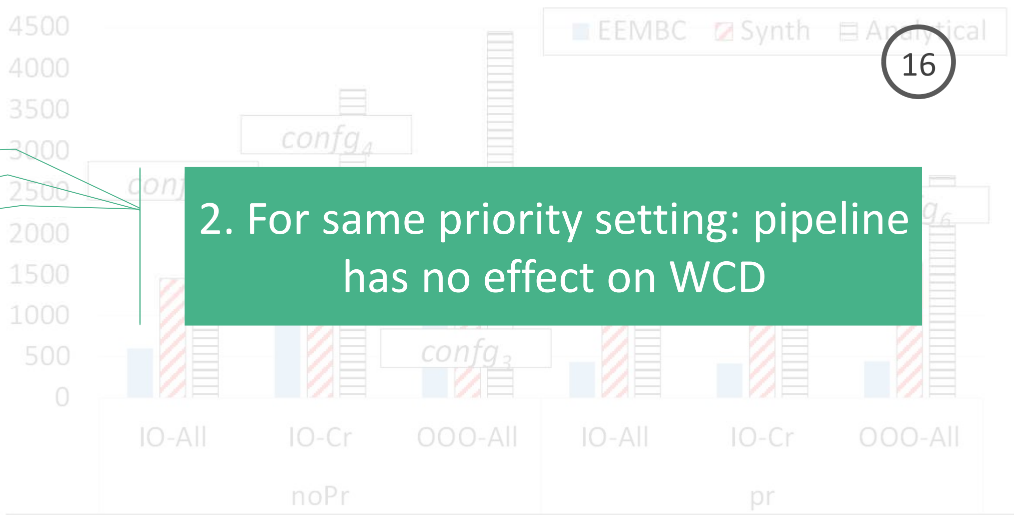
No bank sharing (*Part-All*)

1. Intra-bank reorder (FRFCFS) has no effect on WCD

WCD of Critical Processors

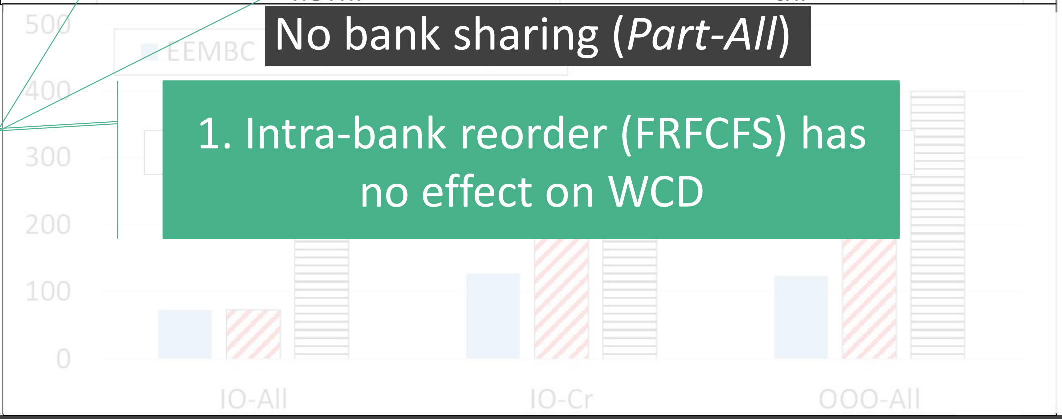
RESULTS



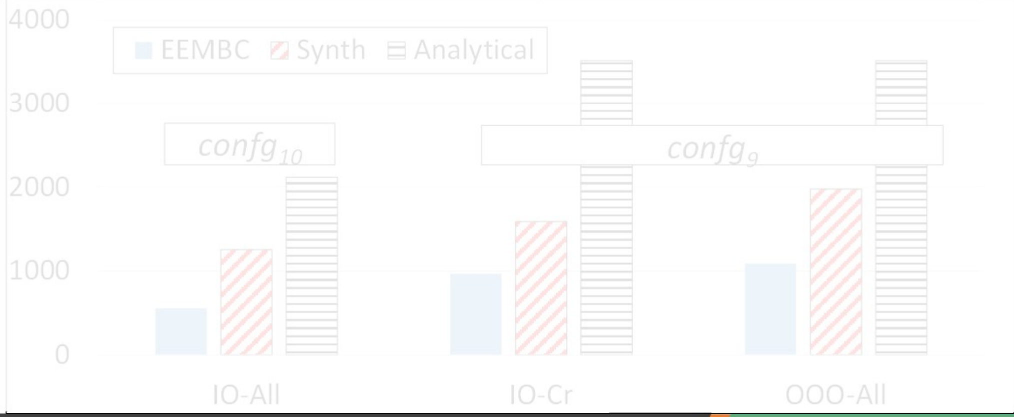


2. For same priority setting: pipeline has no effect on WCD

No bank sharing (*Part-All*)

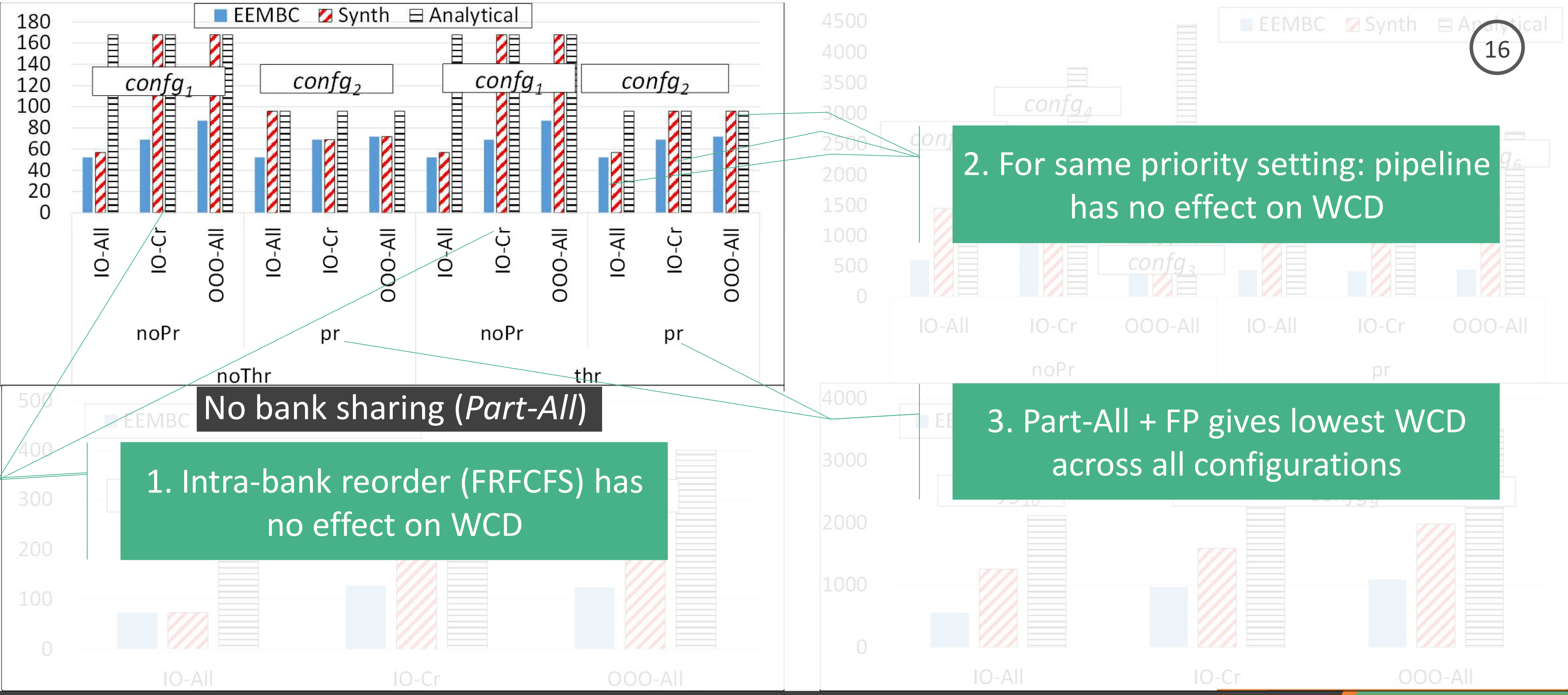


1. Intra-bank reorder (FRFCFS) has no effect on WCD



WCD of Critical Processors

RESULTS



No bank sharing (*Part-All*)

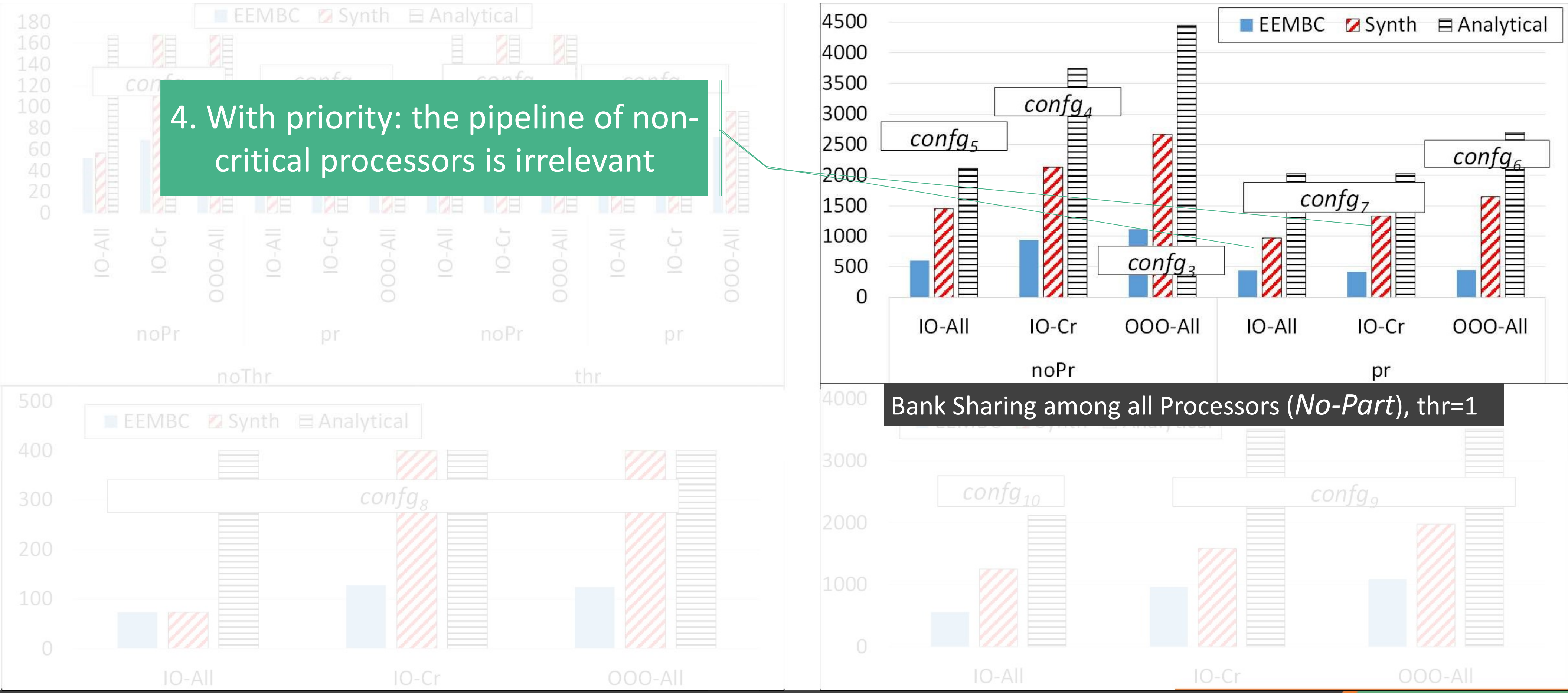
1. Intra-bank reorder (FRFCFS) has no effect on WCD

2. For same priority setting: pipeline has no effect on WCD

3. Part-All + FP gives lowest WCD across all configurations

WCD of Critical Processors

RESULTS

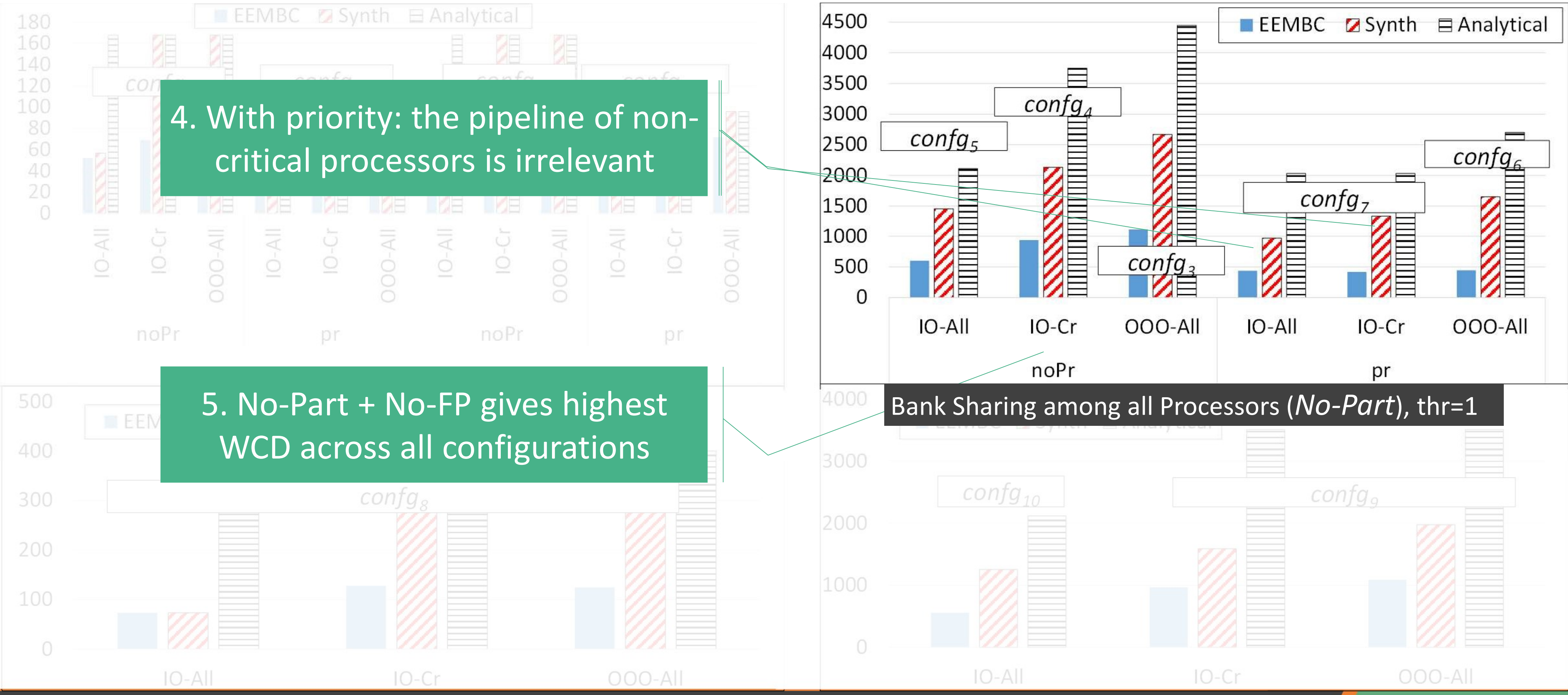


4. With priority: the pipeline of non-critical processors is irrelevant

Bank Sharing among all Processors (*No-Part*), thr=1

WCD of Critical Processors

RESULTS



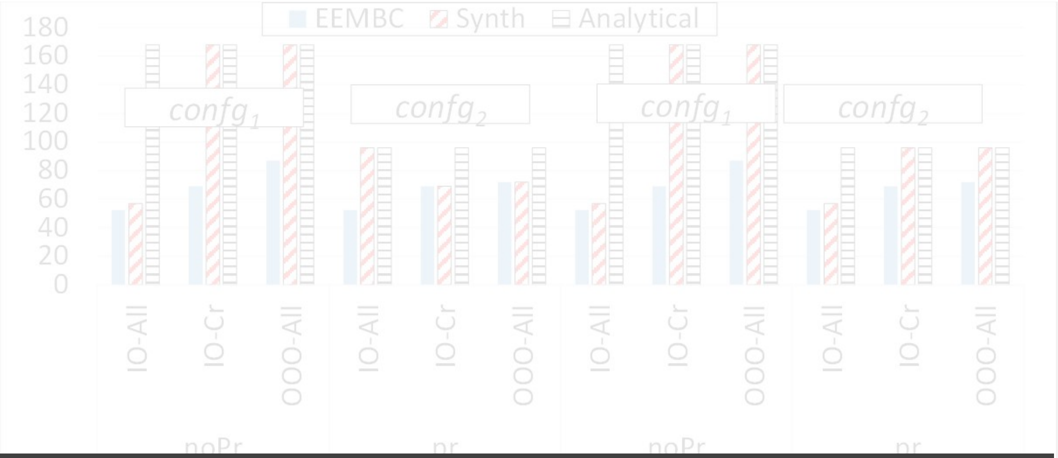
4. With priority: the pipeline of non-critical processors is irrelevant

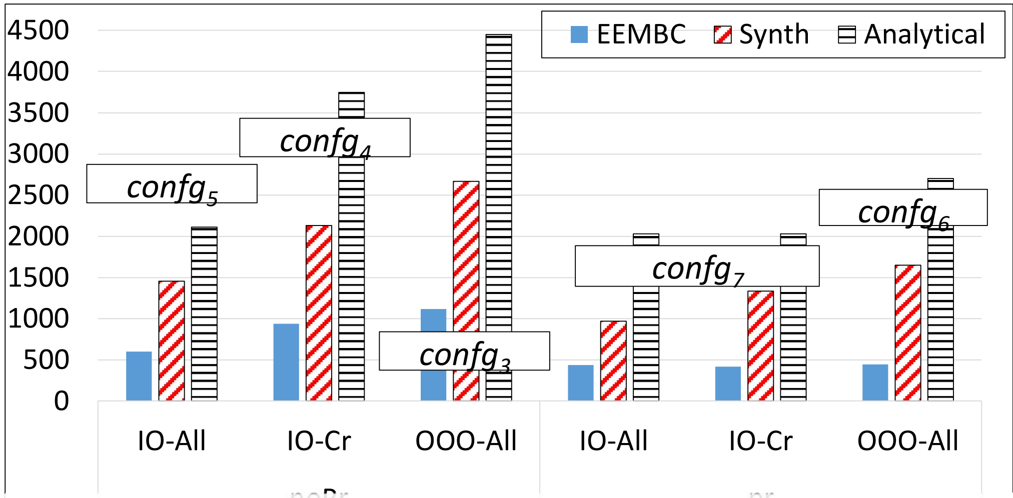
5. No-Part + No-FP gives highest WCD across all configurations

Bank Sharing among all Processors (*No-Part*), thr=1

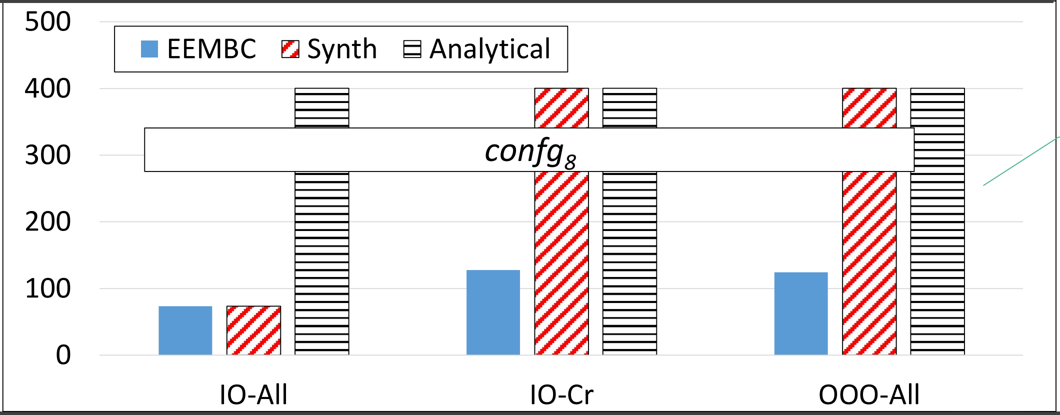
WCD of Critical Processors

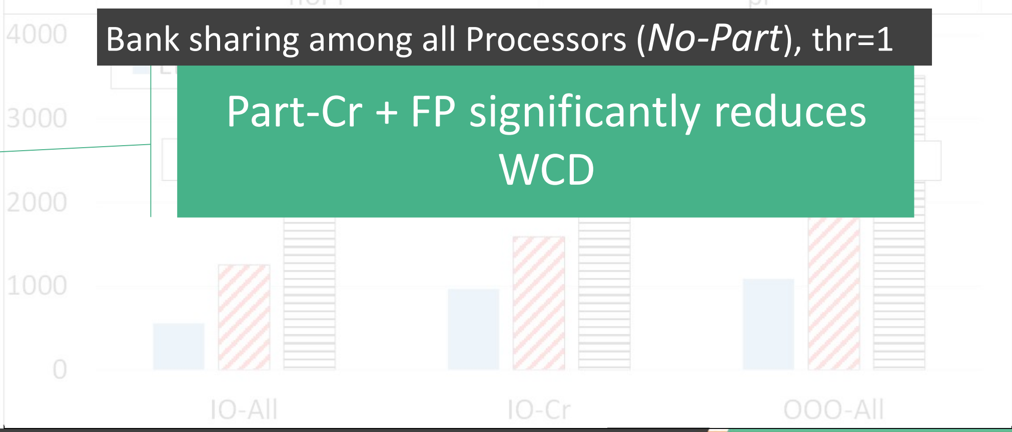
RESULTS





No bank sharing among Critical Processors (Part-Cr), FP





Bank sharing among all Processors (*No-Part*), thr=1

Part-Cr + FP significantly reduces WCD

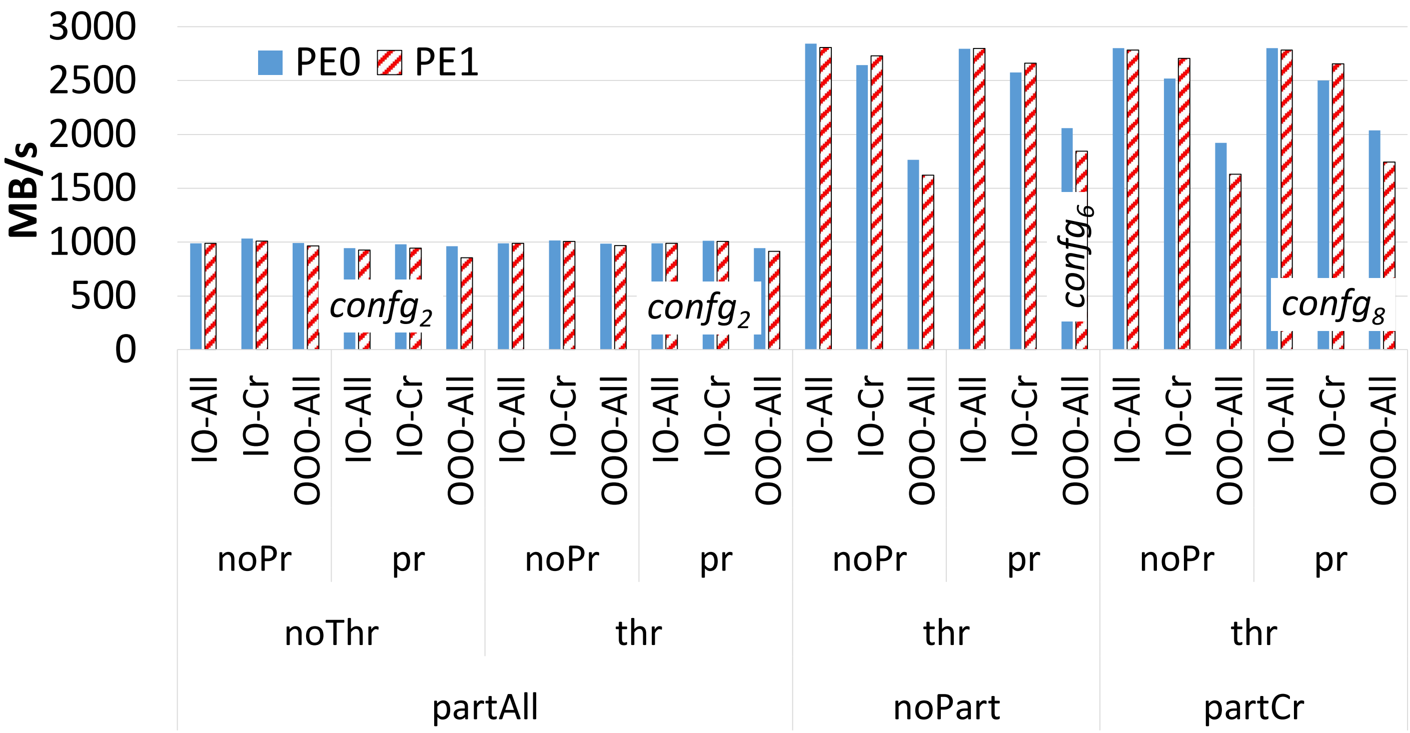
WCD of Critical Processors

RESULTS

Compared to Confg 6 (No-Part):

- Confg 2 (Part-All):
  - 96% less WCD
  - 60% BW degradation

- Confg 8 (Part-Cr + FP):
  - 89% less WCD
  - 0.85% BW degradation



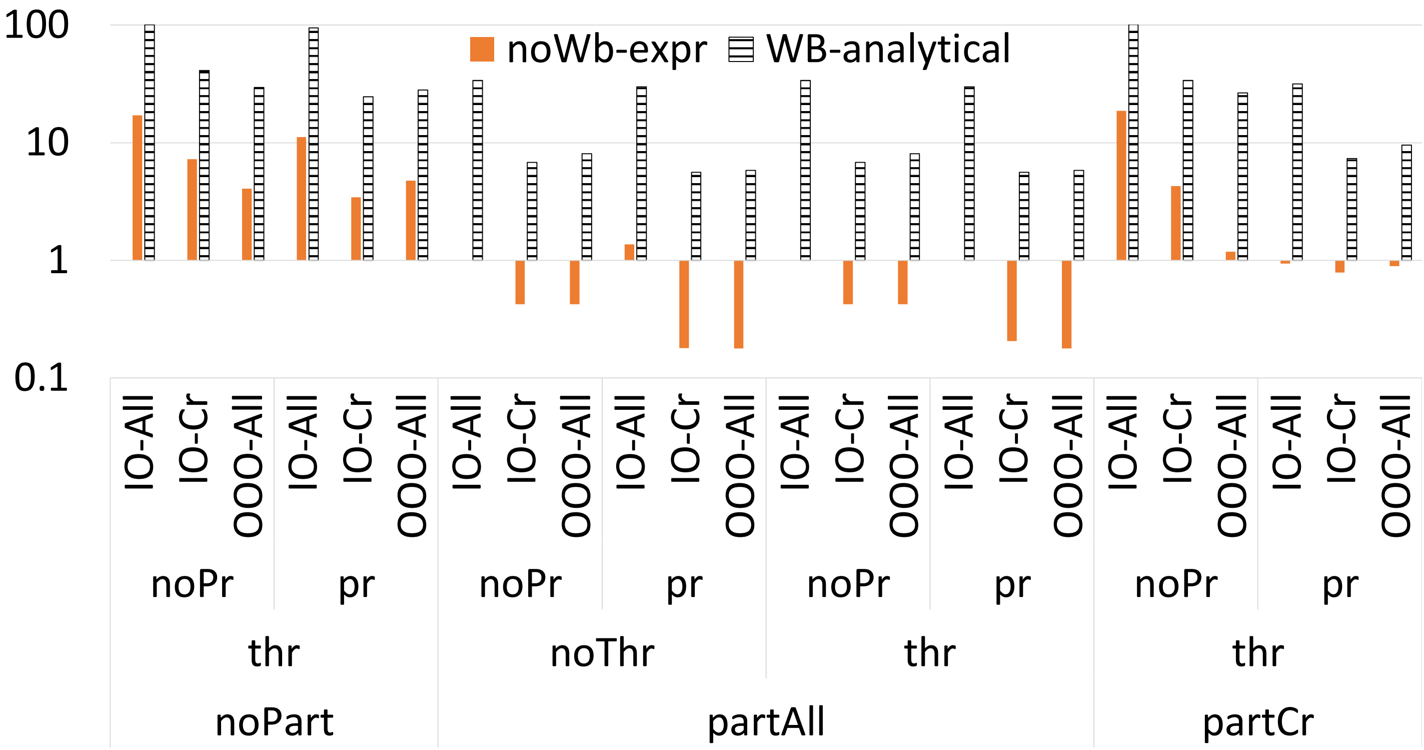
Bandwidth

RESULTS

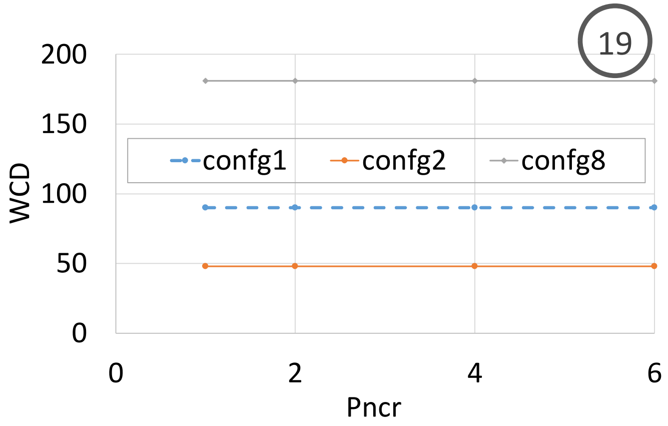
Write Batching Effect

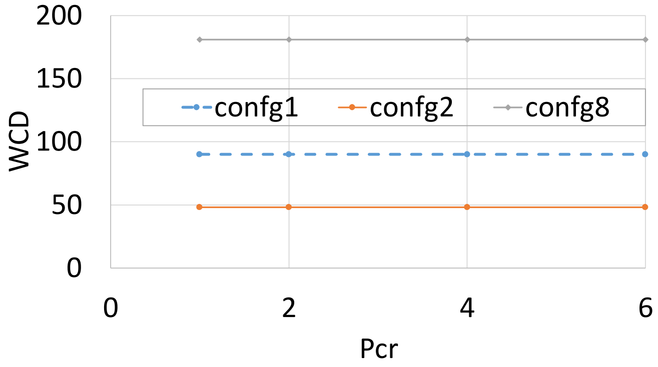
RESULTS

- Normalized to WB-expr
- WB-analytical is very pessimistic
- WB improves avg case
  - noWb-expr is 2.84x on average as compared to Wb-expr
  - even reaches 10x



- Confg. 1 &2 & 8 offers complete isolation



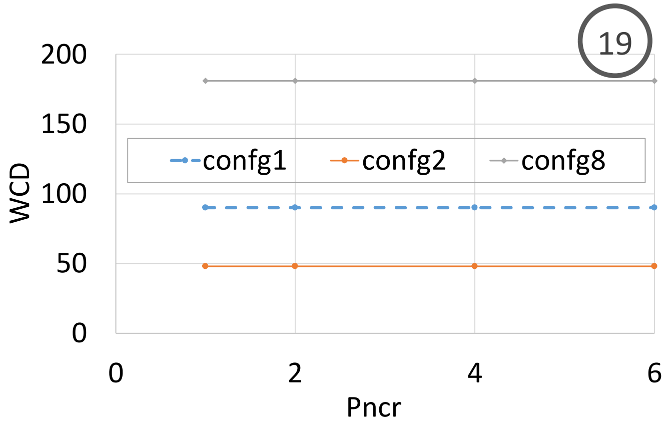


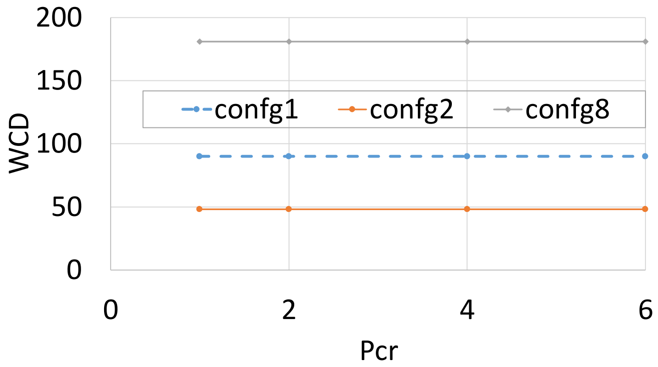
Sensitivity to # Processors

RESULTS

- Confg. 1 &2 & 8 offers complete isolation

- Confg. 1 and 2: *Part-All*
- Confg. 8: *Part-Cr with fixed priority*





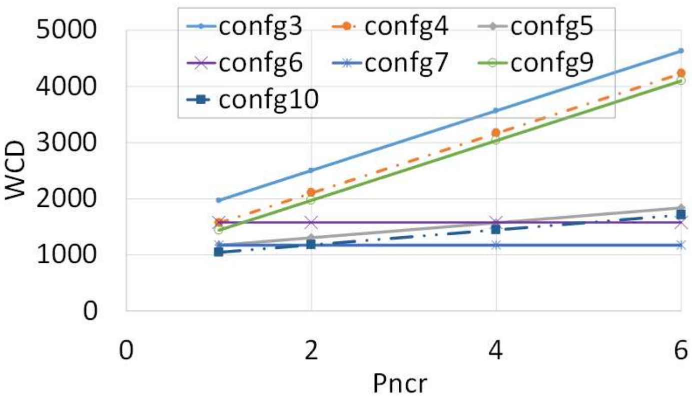
Sensitivity to # Processors

RESULTS

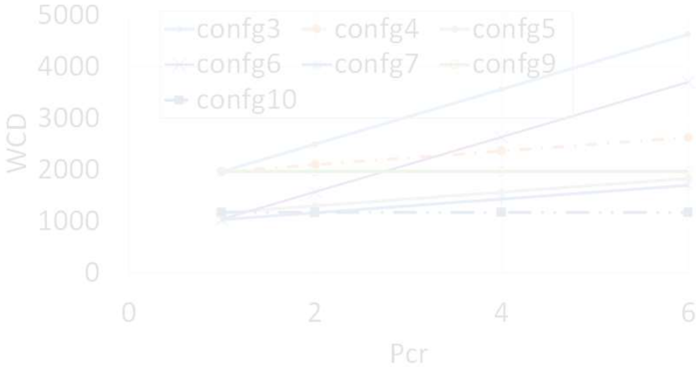
Sensitivity to # Processors

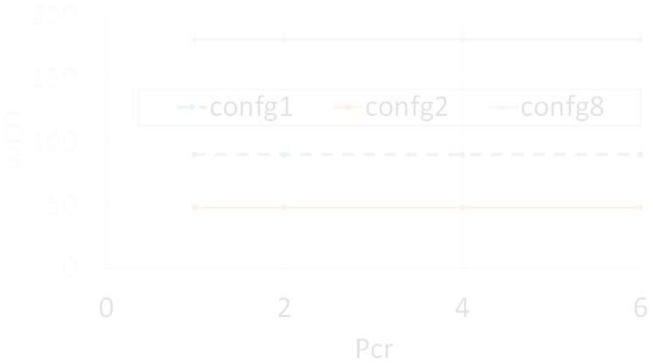
RESULTS

- Confg 1 &2 & 8 offers complete isolation
- Confg 6 & 7 offers isolation from non-cr PEs



- Confg. 6 & 7: deploy *fixed priority*

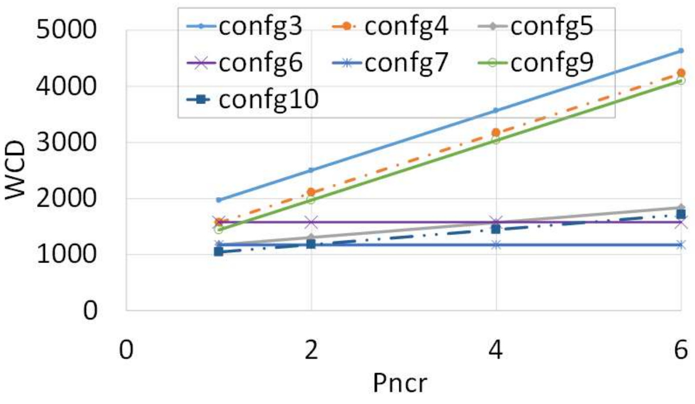




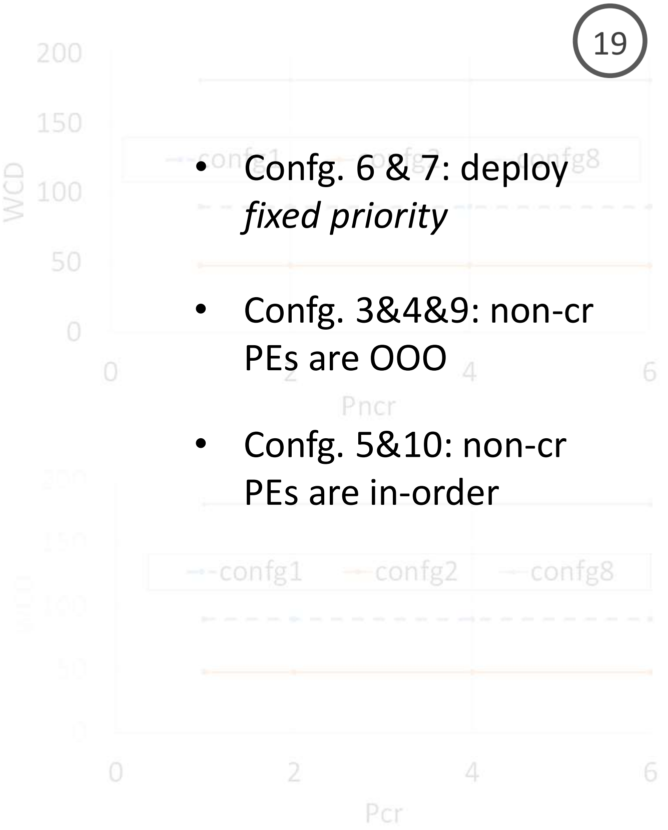
Sensitivity to # Processors

RESULTS

- Confg 1 &2 & 8 offers complete isolation
- Confg 6 & 7 offers isolation from non-cr PEs
- Confg 3,4,9 are more vulnerable to WCD↑ when Pncr↑



confg3
confg4
confg5
confg6
confg7
confg9
confg10
WCD
5000
4000
3000
2000
1000
0
0
2
4
6
Pcr

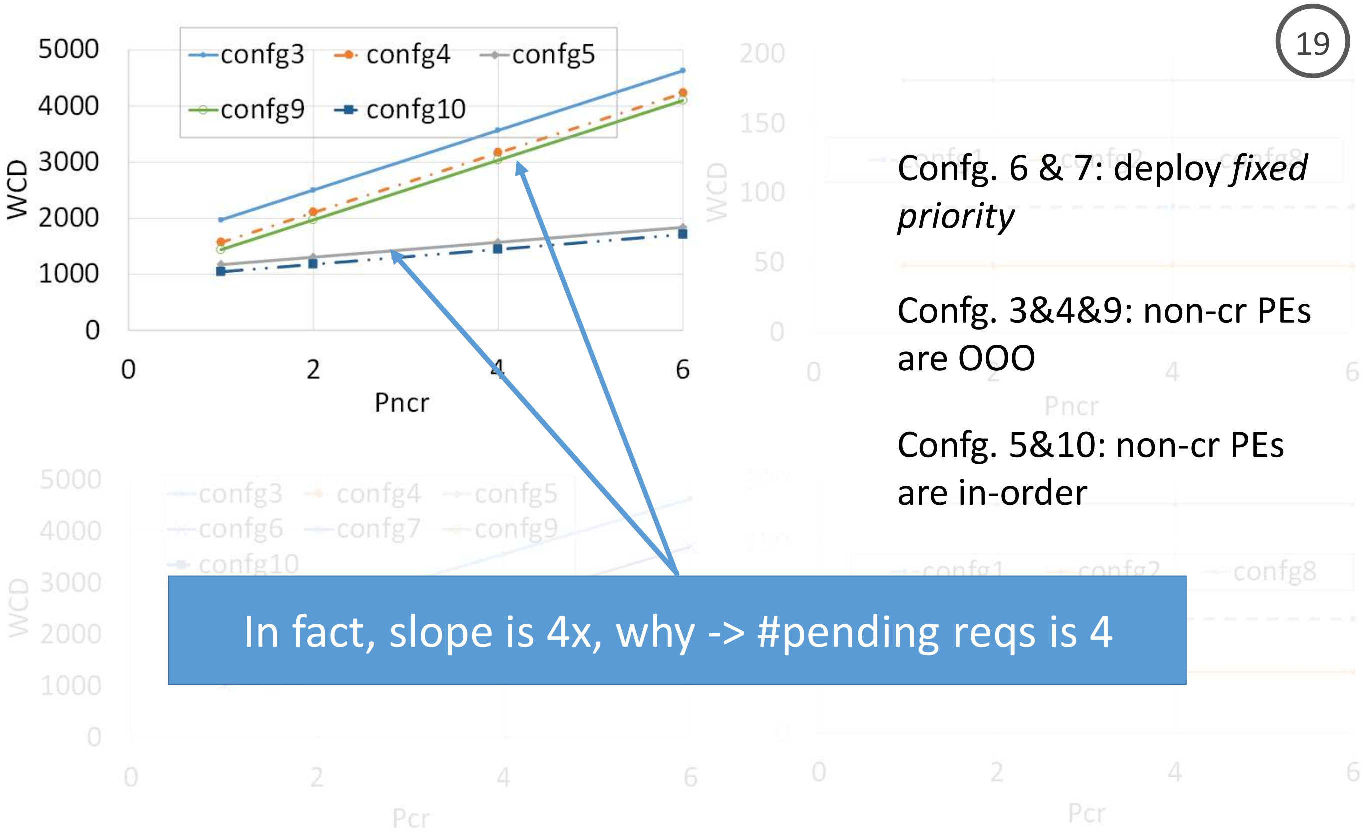


- Confg. 6 & 7: deploy *fixed priority*
- Confg. 3&4&9: non-cr PEs are OOO
- Confg. 5&10: non-cr PEs are in-order

Sensitivity to # Processors

RESULTS

- Confg 1 &2 & 8 offers complete isolation
- Confg 6 & 7 offers isolation from non-cr PEs
- Confg 3,4,9 are more vulnerable to WCD↑ when Pncr↑



Confg. 6 & 7: deploy *fixed priority*

Confg. 3&4&9: non-cr PEs are OOO

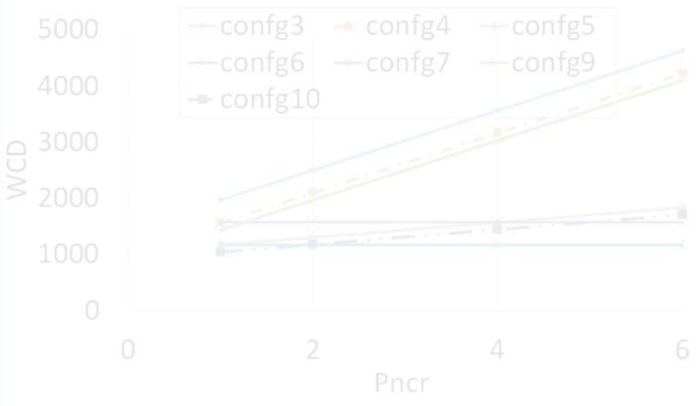
Confg. 5&10: non-cr PEs are in-order

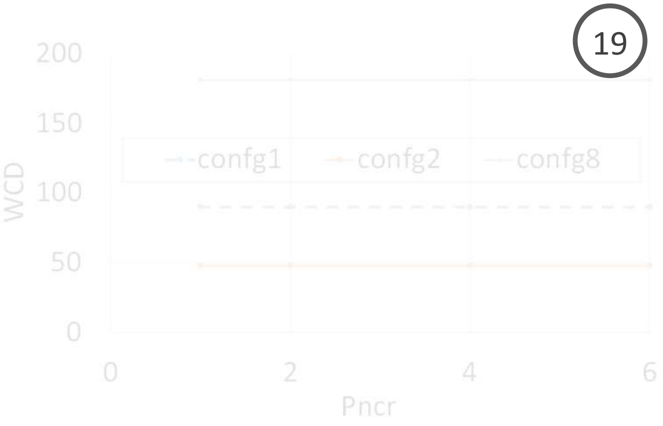
In fact, slope is 4x, why -> #pending reqs is 4

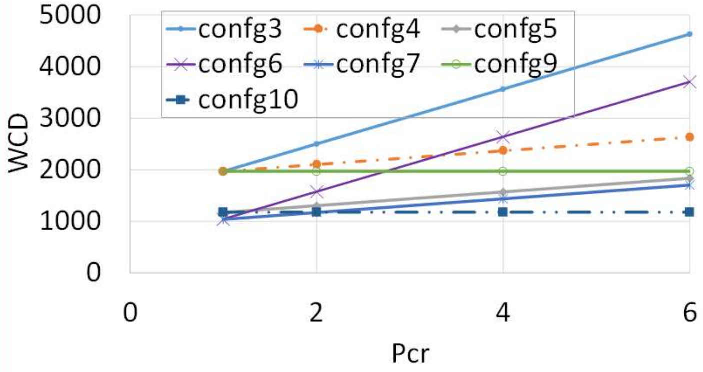
Sensitivity to # Processors

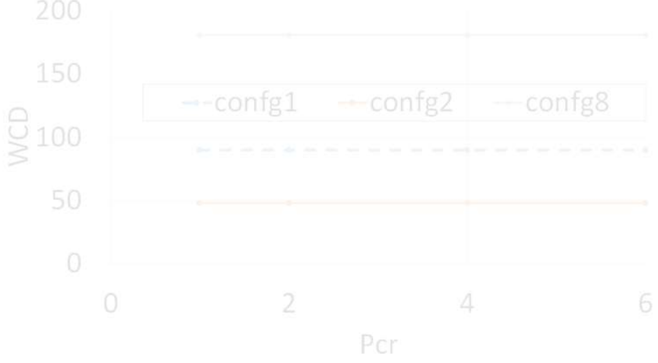
RESULTS

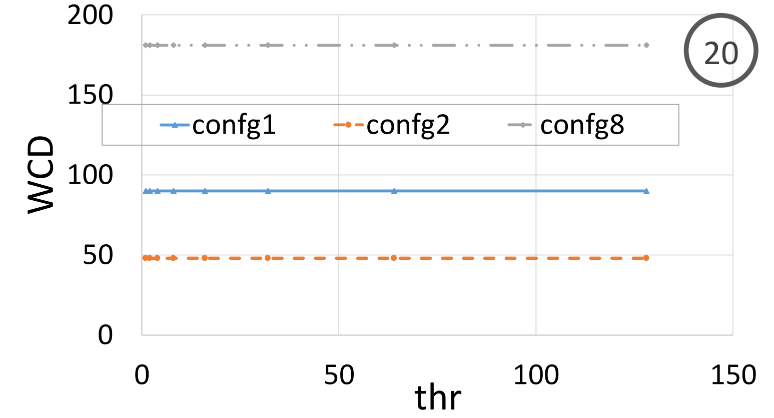
- Confg 1 &2 & 8 offers complete isolation
- Confg 6 & 7 offers isolation from non-cr PEs
- Confg 9 & 10 offers isolation from cr PEs
- Confg 3,4,9 are more vulnerable to WCD↑ when Pncr↑
- Confg 3,6 are more vulnerable to WCD↑ when Pcr ↑

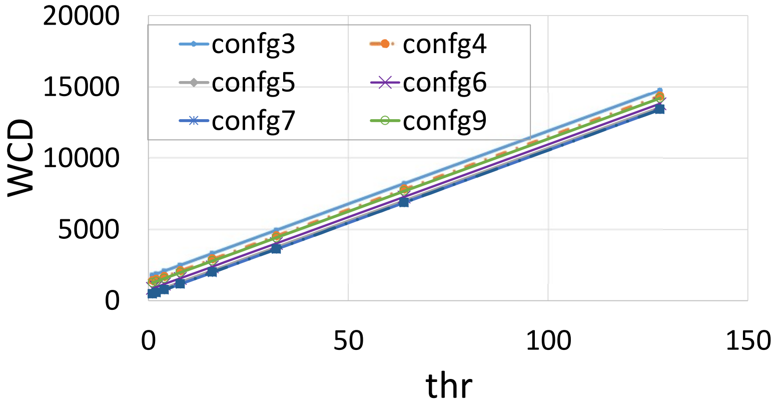












Sensitivity to FR-FCFS thr.

RESULTS

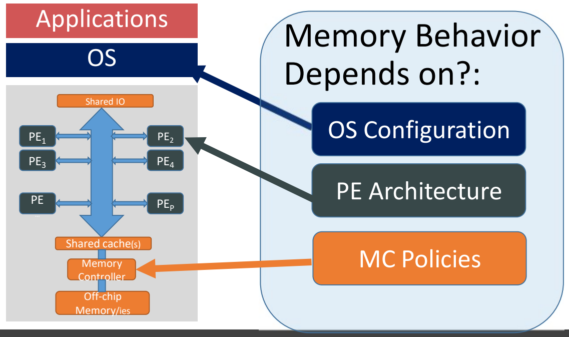
- Heterogeneous MPSoCs are important for Mixed Criticality Systems

Summary & Conclusions

- Heterogeneous MPSoCs are important for Mixed Criticality Systems
- We derived a generalized analysis that bounds the per-request DRAM interference delay in MPSoCs

Summary & Conclusions

- Heterogeneous MPSoCs are important for Mixed Criticality Systems
- We derived a generalized analysis that bounds the per-request DRAM interference delay in MPSoCs



Memory Behavior Depends on?:

OS Configuration

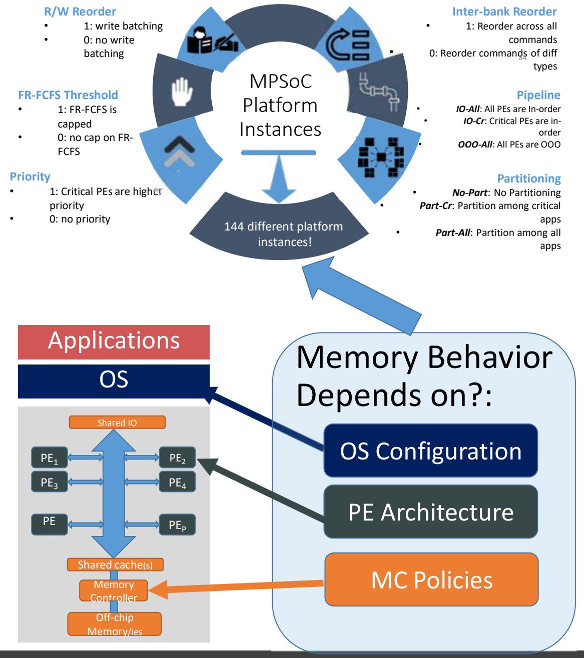
PE Architecture

MC Policies

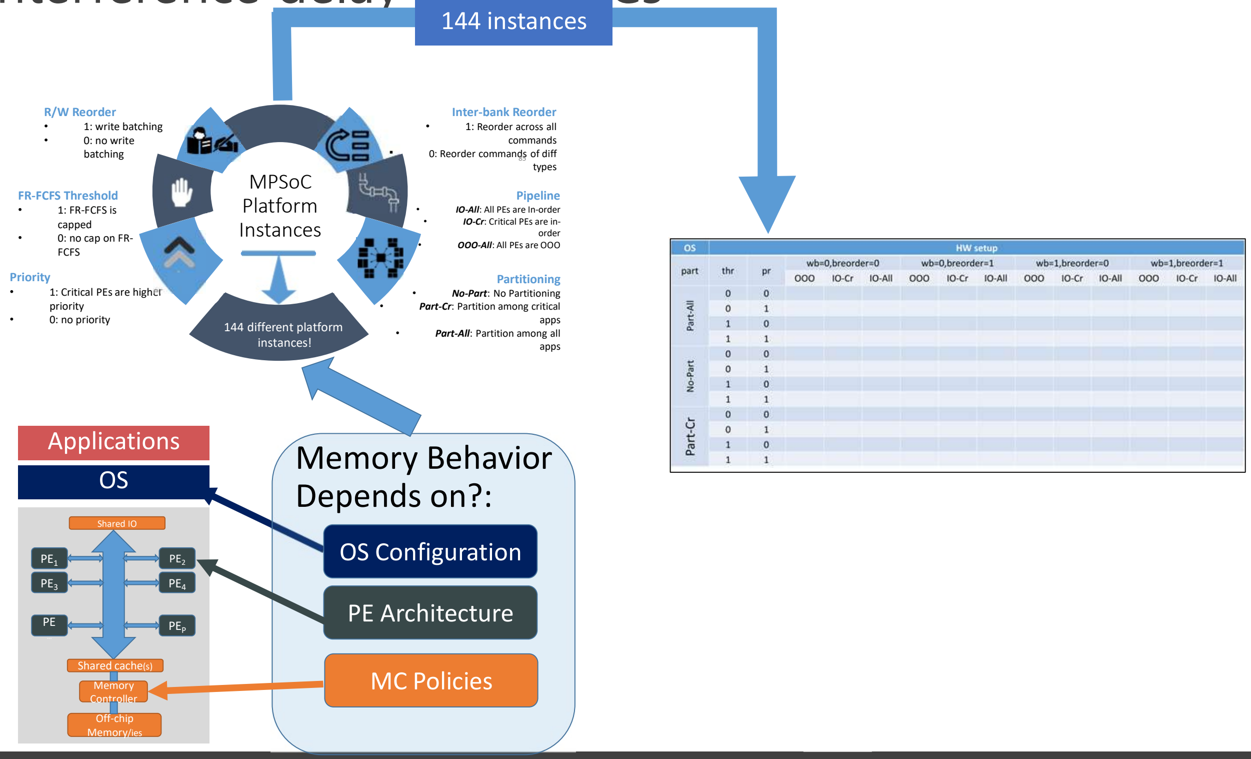
Summary & Conclusions

Summary & Conclusions

- We derived a generalized analysis that bounds the per-request DRAM interference delay in MPSoCs



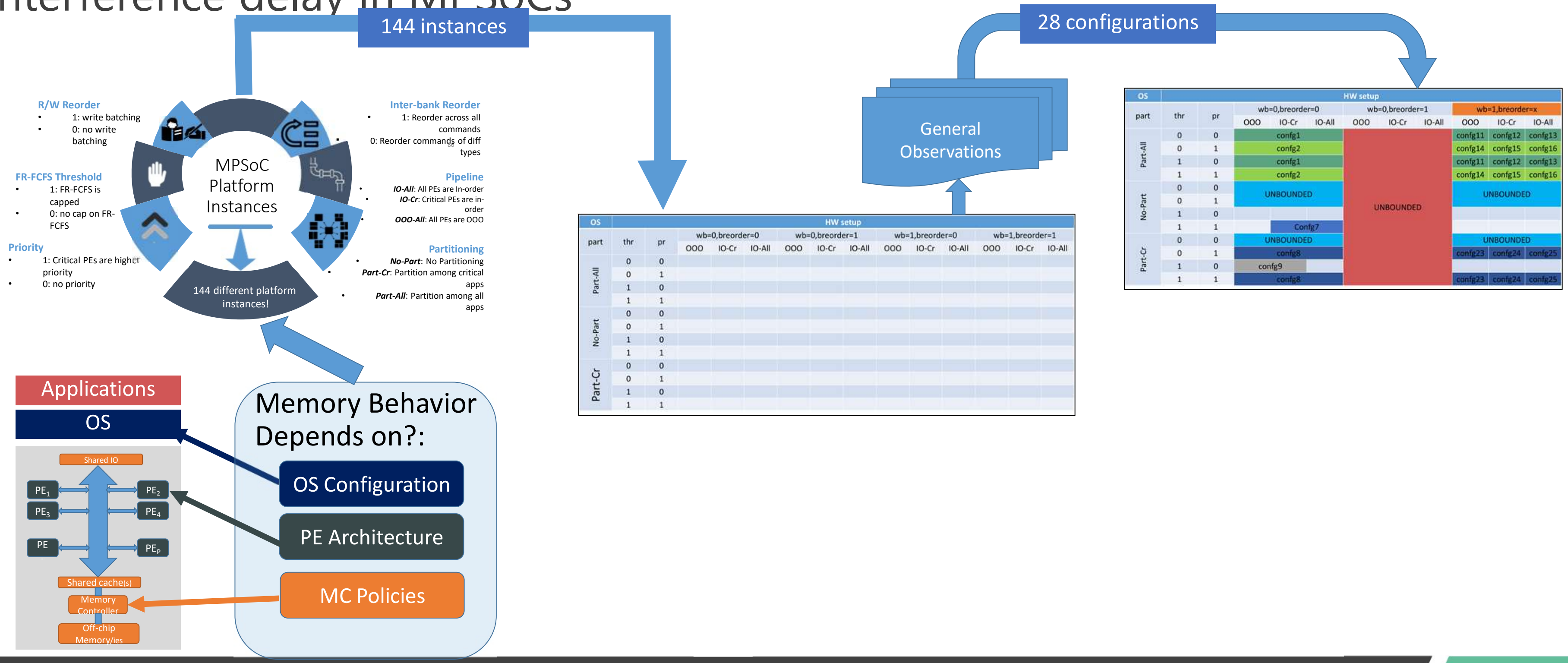
- We derived a generalized analysis that bounds the per-request DRAM interference delay in MPSoCs



OS	HW setup													
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0												
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

Summary & Conclusions

- We derived a generalized analysis that bounds the per-request DRAM interference delay in MPSoCs



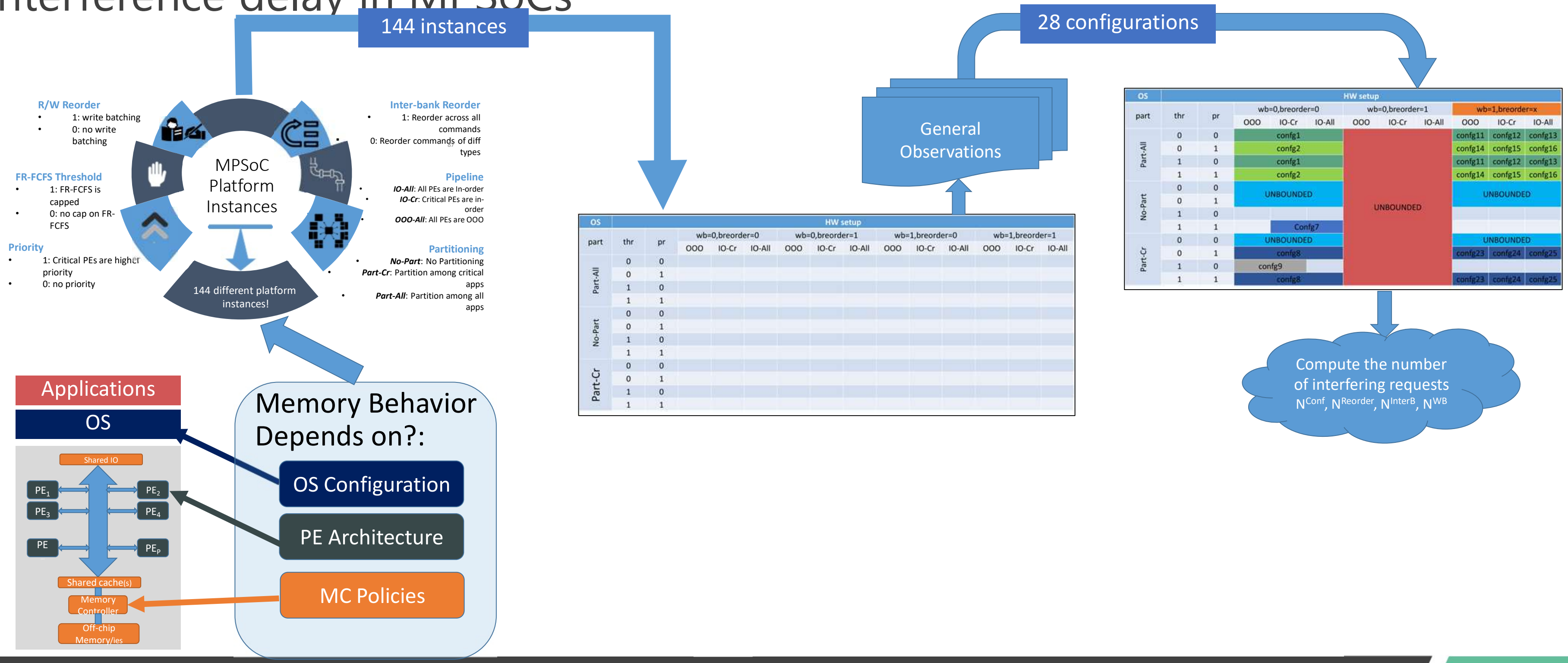
OS			HW setup											
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0												
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

OS			HW setup								
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1			UNBOUNDED			confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1	1	confg2						confg14	confg15	confg16
No-Part	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1		Confg7							
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1	confg8						confg23	confg24	confg25
	1	0	confg9								
	1	1	confg8						confg23	confg24	confg25

Summary & Conclusions

Summary & Conclusions

- We derived a generalized analysis that bounds the per-request DRAM interference delay in MPSoCs



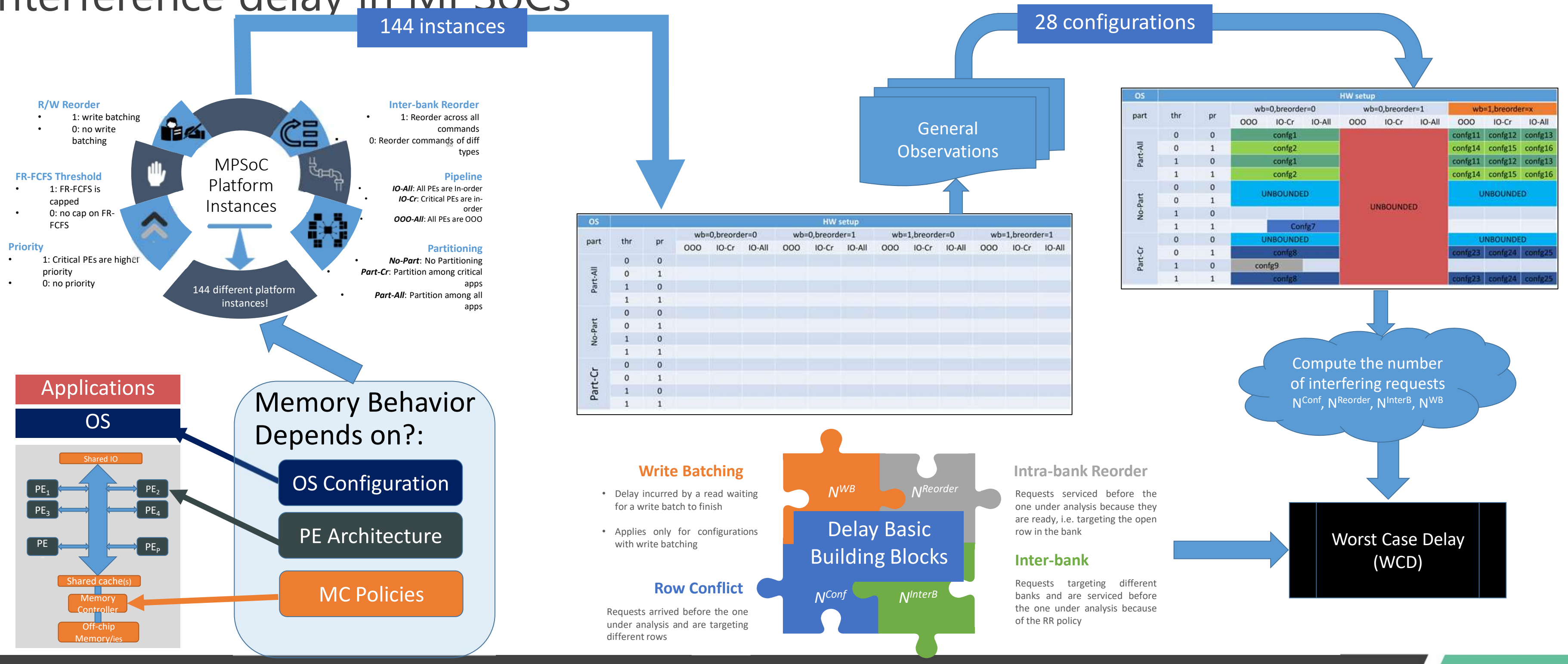
OS			HW setup											
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0												
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

OS			HW setup								
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1			UNBOUNDED			confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1	1	confg2						confg14	confg15	confg16
No-Part	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1		Confg7							
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1	confg8						confg23	confg24	confg25
	1	0	confg9								
	1	1	confg8						confg23	confg24	confg25

N^{Conf}, $N^{Reorder}$, N^{InterB}, N^{WB}

Summary & Conclusions

- We derived a generalized analysis that bounds the per-request DRAM interference delay in MPSoCs



R/W Reorder

- 1: write batching
- 0: no write batching

FR-FCFS Threshold

- 1: FR-FCFS is capped
- 0: no cap on FR-FCFS

Priority

- 1: Critical PEs are higher priority
- 0: no priority

Inter-bank Reorder

- 1: Reorder across all commands
- 0: Reorder commands of diff types

Pipeline

- ***IO-All***: All PEs are In-order
- ***IO-Cr***: Critical PEs are in-order
- ***OOO-All***: All PEs are OOO

Partitioning

- ***No-Part***: No Partitioning
- ***Part-Cr***: Partition among critical apps
- ***Part-All***: Partition among all apps

Memory Behavior Depends on?:

- OS Configuration
- PE Architecture
- MC Policies

OS	HW setup													
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=0			wb=1,breorder=1		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0												
	0	1												
	1	0												
	1	1												
No-Part	0	0												
	0	1												
	1	0												
	1	1												
Part-Cr	0	0												
	0	1												
	1	0												
	1	1												

OS	HW setup										
part	thr	pr	wb=0,breorder=0			wb=0,breorder=1			wb=1,breorder=x		
			OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All	OOO	IO-Cr	IO-All
Part-All	0	0	confg1			UNBOUNDED			confg11	confg12	confg13
	0	1	confg2						confg14	confg15	confg16
	1	0	confg1						confg11	confg12	confg13
	1	1	confg2						confg14	confg15	confg16
No-Part	0	0	UNBOUNDED						UNBOUNDED		
	0	1									
	1	0									
	1	1		Confg7							
Part-Cr	0	0	UNBOUNDED						UNBOUNDED		
	0	1	confg8						confg23	confg24	confg25
	1	0	confg9								
	1	1	confg8						confg23	confg24	confg25

Compute the number of interfering requests N^{Conf}, $N^{Reorder}$, N^{InterB}, N^{WB}

Write Batching

- Delay incurred by a read waiting for a write batch to finish
- Applies only for configurations with write batching

Row Conflict

Requests arrived before the one under analysis and are targeting different rows

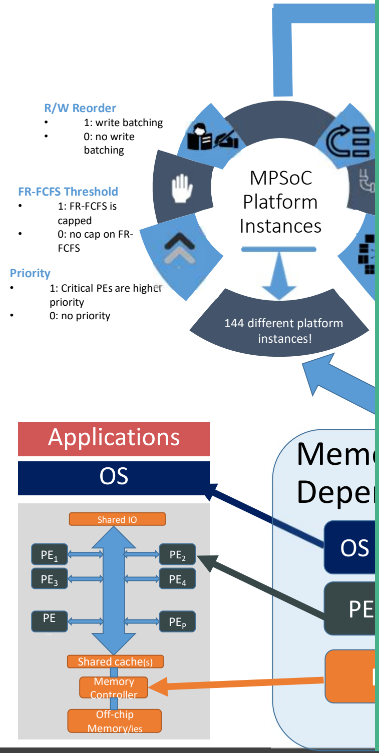
Intra-bank Reorder

Requests serviced before the one under analysis because they are ready, i.e. targeting the open row in the bank

Inter-bank

Requests targeting different banks and are serviced before the one under analysis because of the RR policy

- We derived a ger interference dela

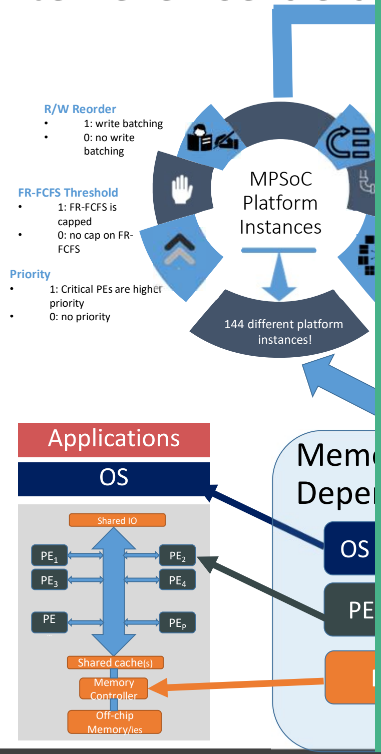


Main lessons:

1. DRAM’s WCD significantly depends on MPSoC features

Summary & Conclusions

- We derived a ger
interference dela

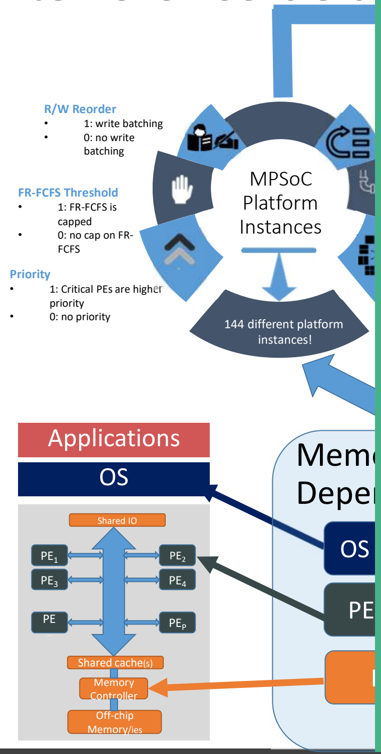


Main lessons:

1. DRAM’s WCD significantly depends on MPSoC features
2. Identified features that lead to unbounded WCD

Summary & Conclusions

- We derived a ger
  interference dela

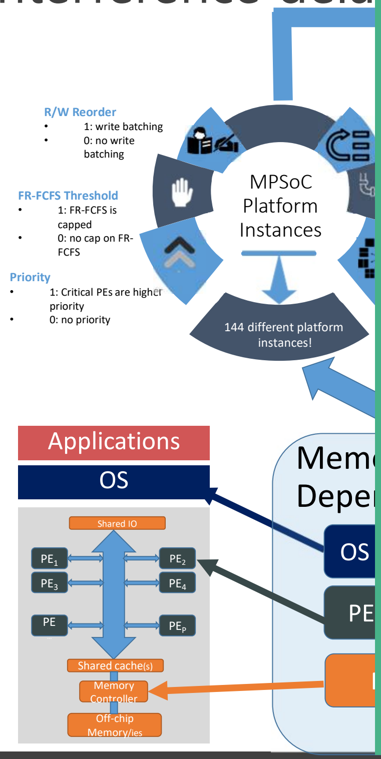


Main lessons:

1. DRAM’s WCD significantly depends on MPSoC features
2. Identified features that lead to unbounded WCD
3. leveraging existing features such as PE prioritization can allow the designer to better trade-off the maximum delay for critical applications and the bandwidth for non-critical ones.

Summary & Conclusions

- We derived a ger interference dela



Main lessons:

1. DRAM’s WCD significantly depends on MPSoC features
2. Identified features that lead to unbounded WCD
3. leveraging existing features such as PE prioritization can allow the designer to better trade-off the maximum delay for critical applications and the bandwidth for non-critical ones.
4. There is interdependency among the effects of the features on both the delay and the bandwidth. Existence of some features can countermand the effect of other features

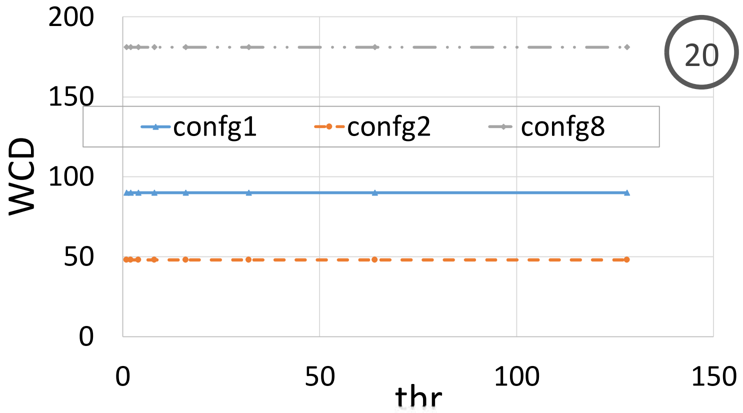
Summary & Conclusions

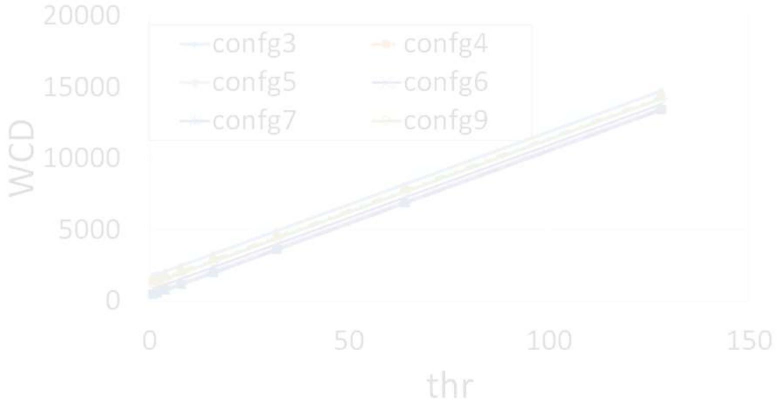
Sensitivity to FR-FCFS thr.

RESULTS

- Confg 1 &2 &8 offers complete isolation from FR-FCFS reordering

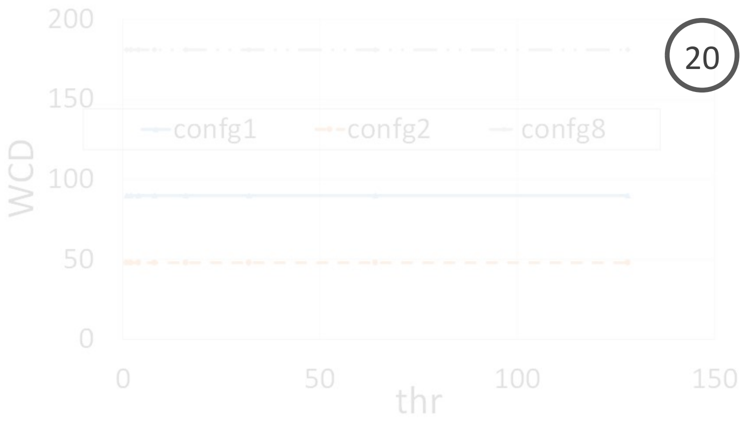
- Confg. 1 and 2: *Part-All*
- Confg. 8: *Part-Cr with fixed priority*

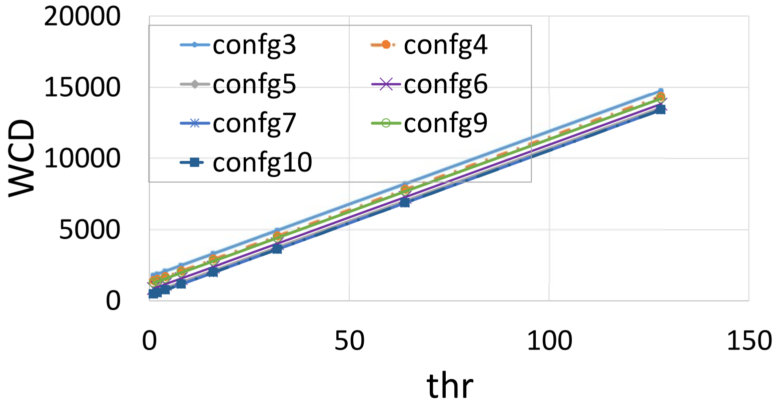




- Confg 1 &2 &8 offers complete isolation from FR-FCFS threshold
- Configs 3-7 & 10 scales linearly with FR-FCFS threshold

- Slope is the same for these configs
- $L^{Reorder}$ component depends only on thr and JEDEC constraints
- Reordering has huge impact on WCD

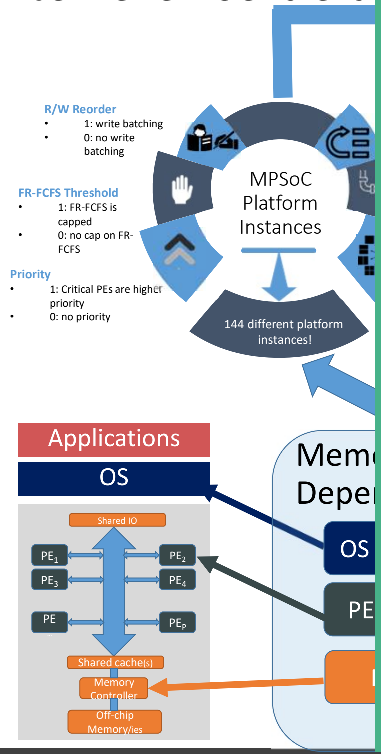




Sensitivity to FR-FCFS thr.

RESULTS

- We derived a ger interference dela



Main lessons:

1. DRAM’s WCD significantly depends on MPSoC features
2. Identified features that lead to unbounded WCD
3. leveraging existing features such as PE prioritization can allow the designer to better trade-off the maximum delay for critical applications and the bandwidth for non-critical ones.
4. There is interdependency among the effects of the features on both the delay and the bandwidth. Existence of some features can countermand the effect of other features
5. Although write batching mechanism works well in the average case, it unfortunately induces pathological cases that result in high bounds on per-request delay

Summary & Conclusions

Main lessons:

1. DRAM’s WCD significantly depends on MPSoC features
2. Identified features that lead to unbounded WCD
3. leveraging existing features such as PE prioritization can allow the designer to better trade-off the maximum delay for critical applications and the bandwidth for non-critical ones.
4. There is interdependency among the effects of the features on both the delay and the bandwidth. Existence of some features can countermand the effect of other features
5. Although write batching mechanism works well in the average case, it unfortunately induces pathological cases that result in high bounds on per-request delay

Summary & Conclusions