FANUS

MOHAMED HASSAN

TOWARDS PREDICTABLE, SECURE, AND VERIFIED CYBER-PHYSICAL SYSTEMS-ON-CHIP (CPSoCs)

A disclaimer: Work presented in this talk has been done while affiliated to one of these fantastic places:

McMaster University

University of Guelph

intel

University of Waterloo

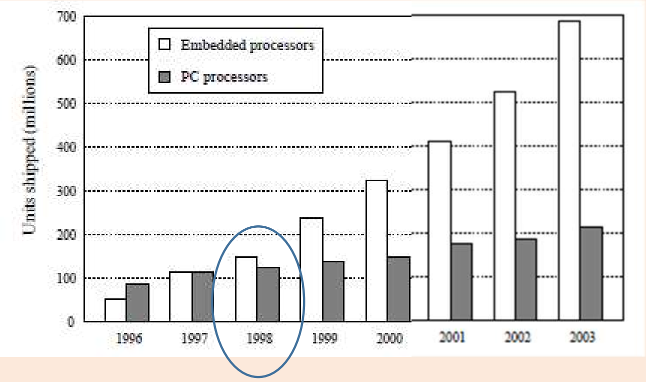
Feb 18th, 2020

Embedded Systems, The Future

MOTIVATION

Embedded Systems, The Future

MOTIVATION

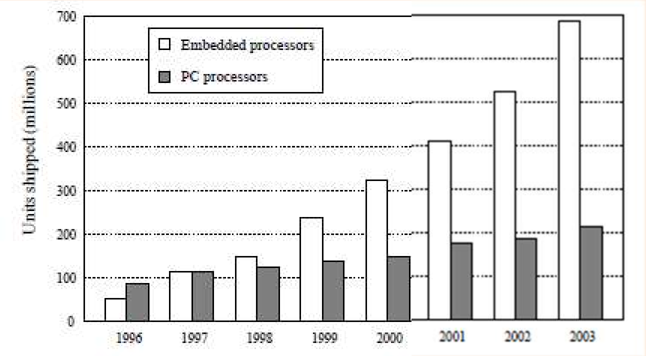


# Embedded processors shipped units already surpassed PCs

1998

Embedded Systems, The Future

MOTIVATION

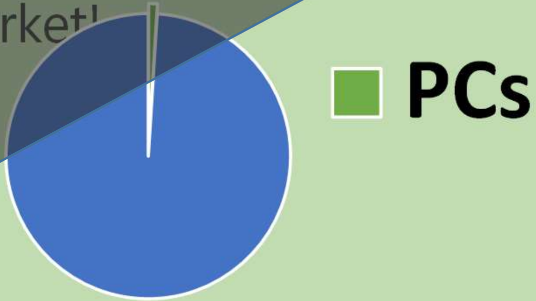


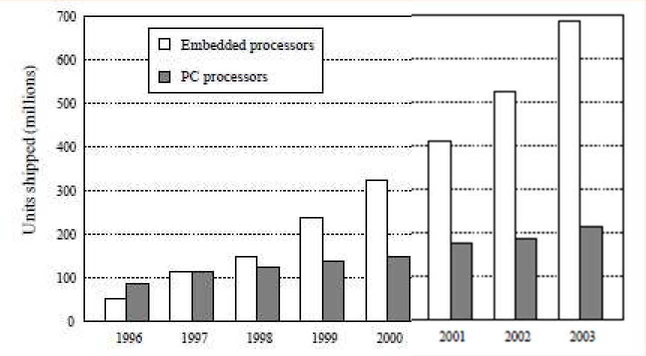
# Embedded processors shipped units already surpassed PCs

1998

Now?

All PCs in the world are less than 1% of the market!



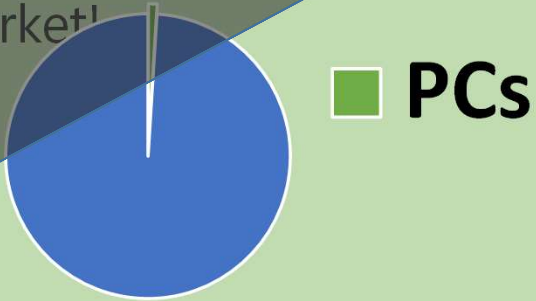


# Embedded processors shipped units already surpassed PCs

1998

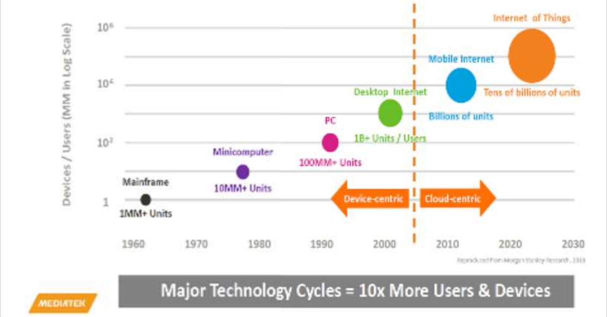
Now?

All PCs in the world are less than 1% of the market!



2030?

- IoT are forecasted to have billions of units by 2030!
- Autonomous cars have 100s of embedded processors

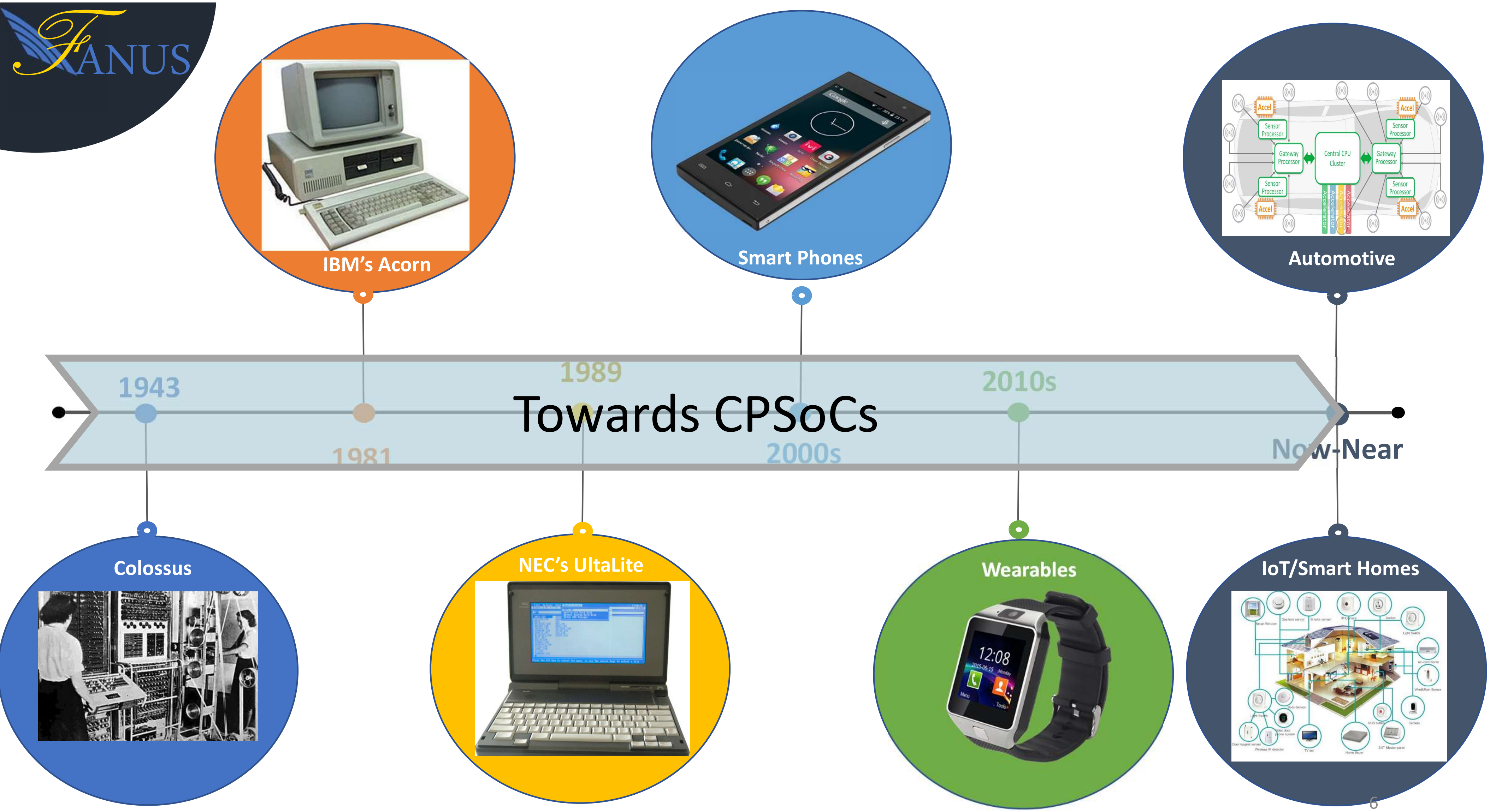


Major Technology Cycles = 10x More Users & Devices

Embedded Systems, The Future

MOTIVATION

Towards CPSoCs



1943

Colossus

1981

IBM's Acorn

1989

NEC's UltaLite

2000s

Smart Phones

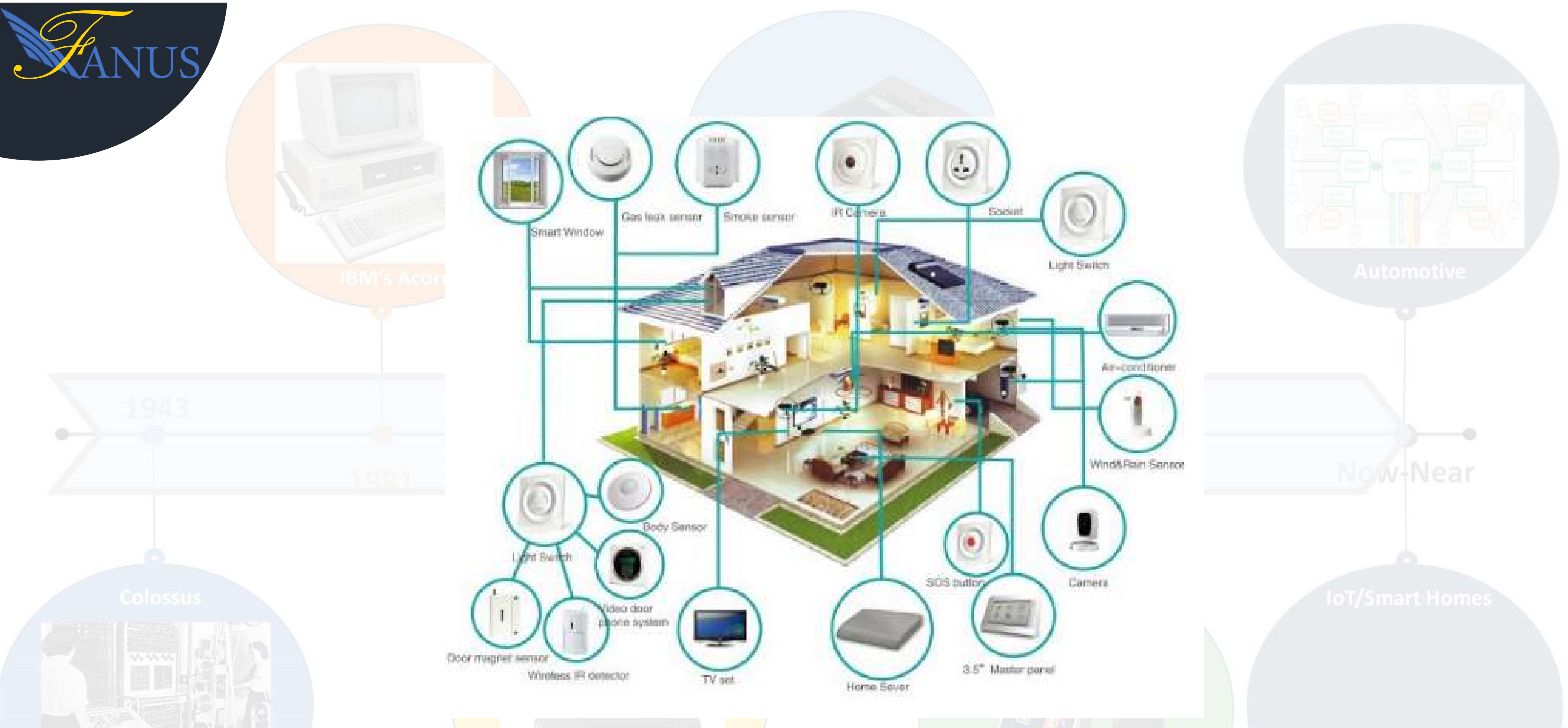
2010s

Wearables

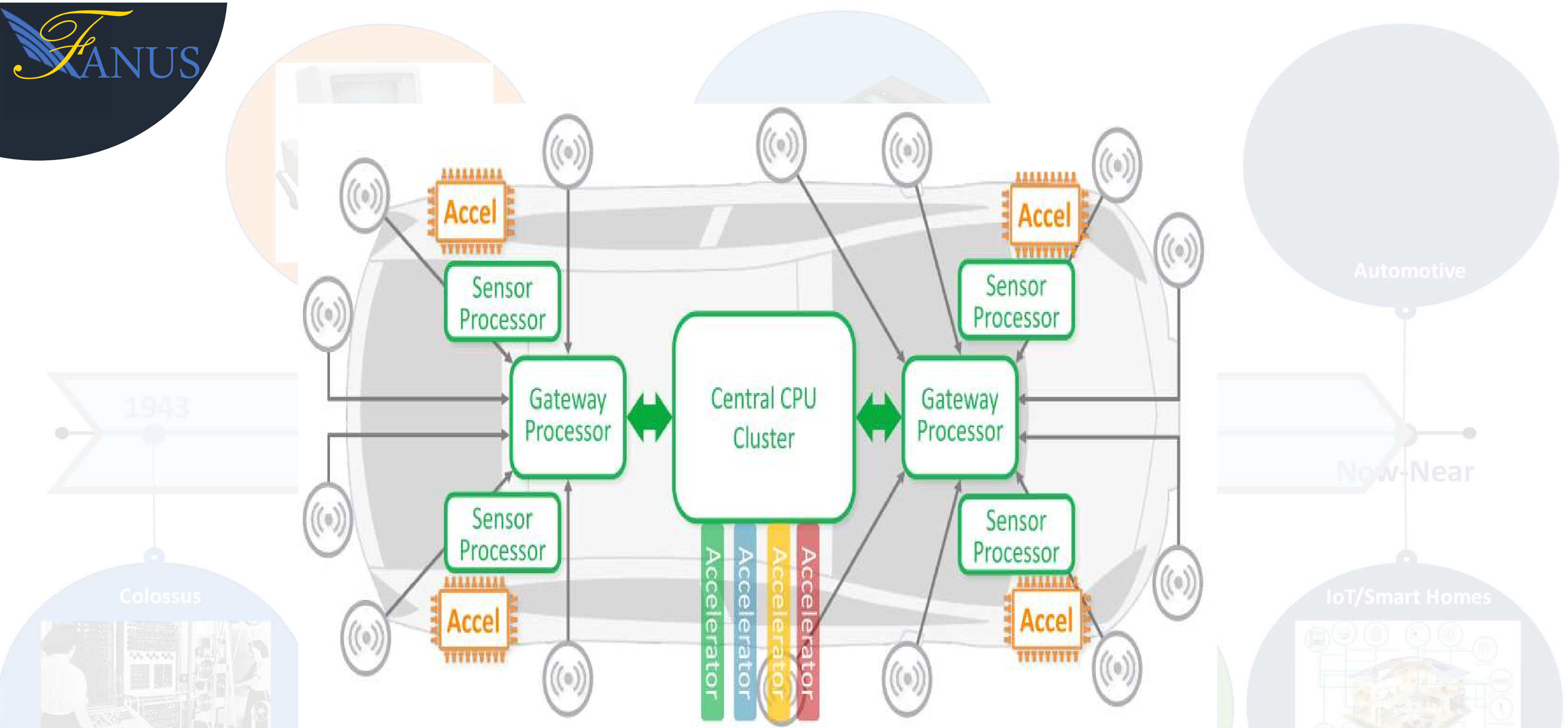
Now-Near

Automotive

IoT/Smart Homes


IBM's Acorn
1943
1981
Colossus
Automotive
Now-Near
IoT/Smart Homes
Smart Window
Gas leak sensor
Smoke sensor
IR Camera
Socket
Light Switch
Air-conditioner
Wind&Rain Sensor
Body Sensor
Light Switch
Video door phone system
Door magnet sensor
Wireless IR detector
TV set
Home Sever
SOS button
Camera
3.5" Master panel

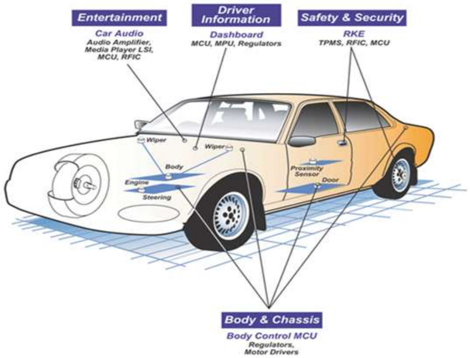
The IoT Case

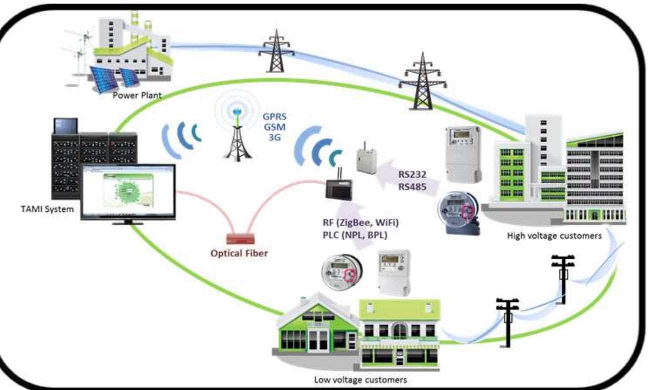


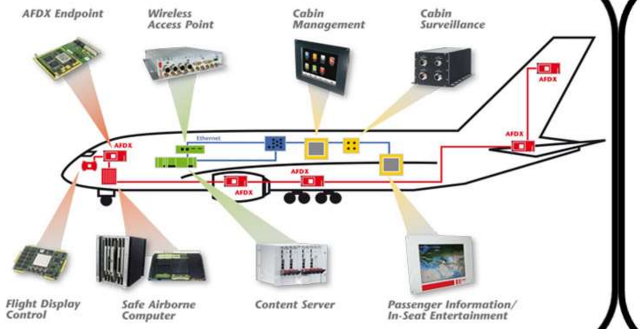
The Automotive Case

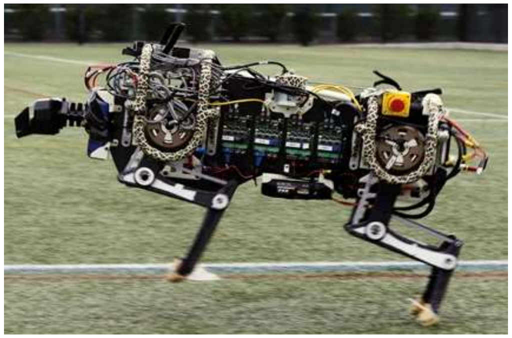
Embedded Systems →Cyber-Physical Systems

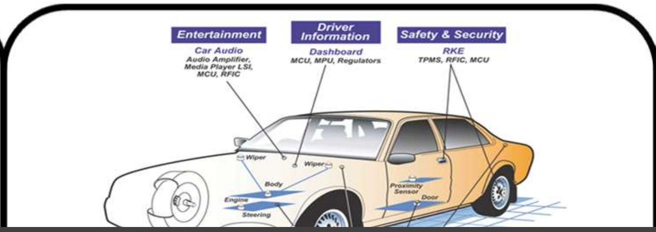
MOTIVATION

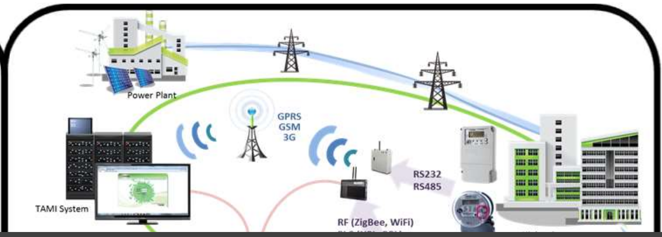












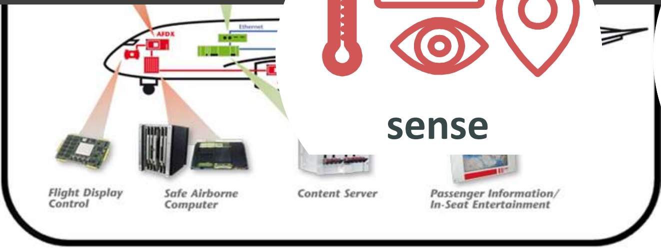
Cyber-Physical Systems

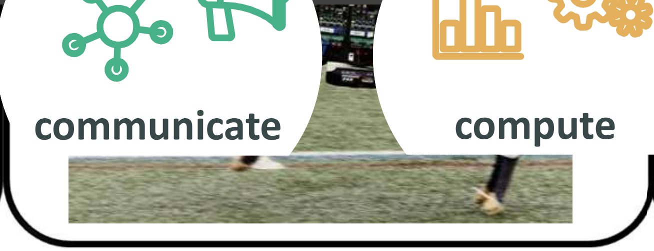
sense

communicate

compute

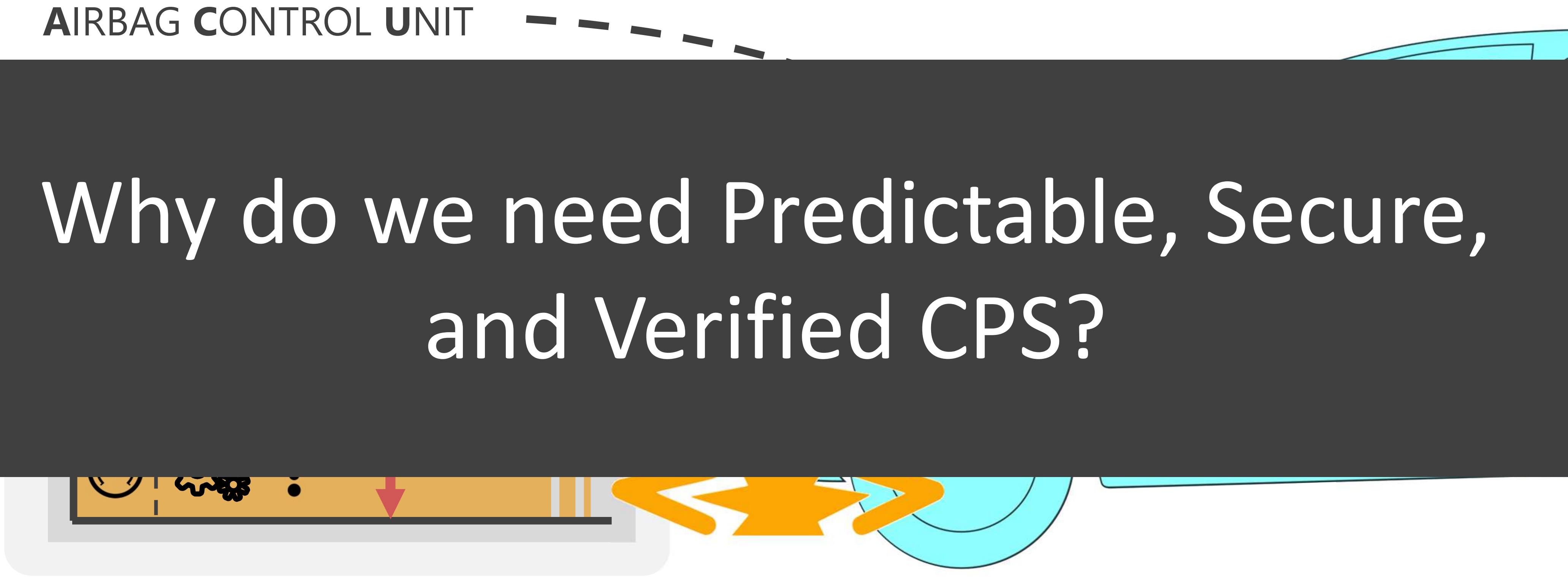
actuate





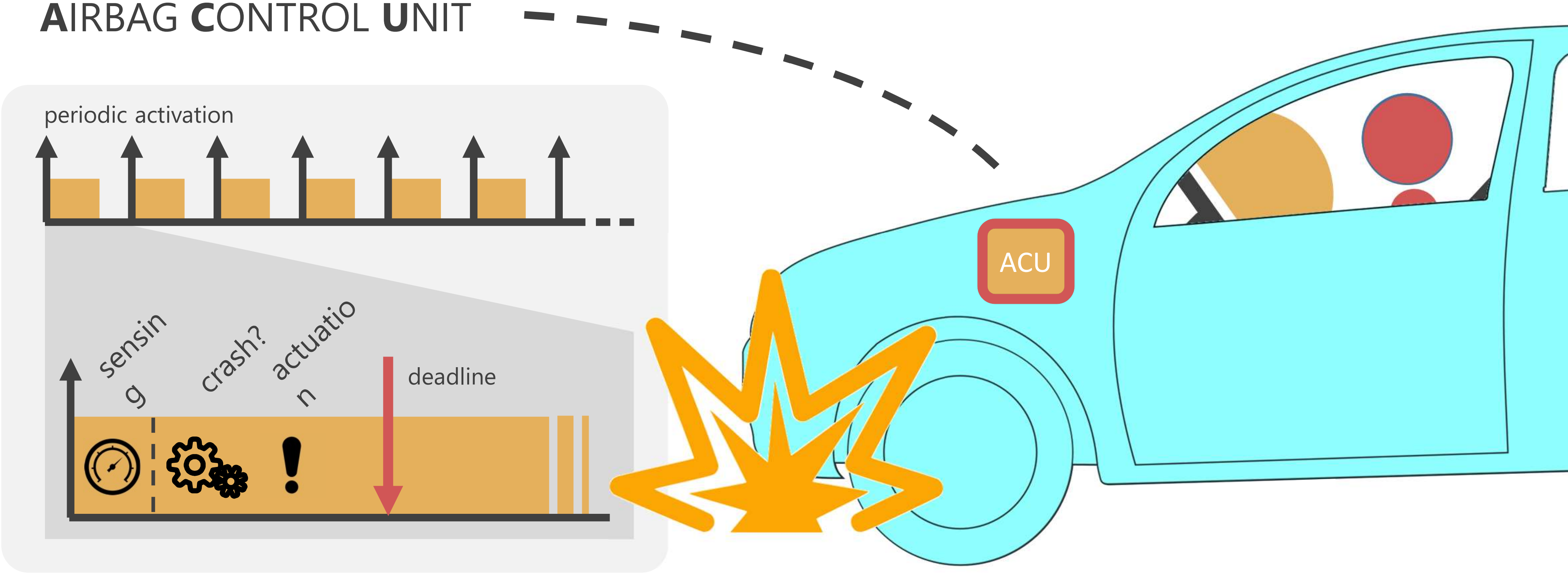
Embedded Systems →Cyber-Physical Systems

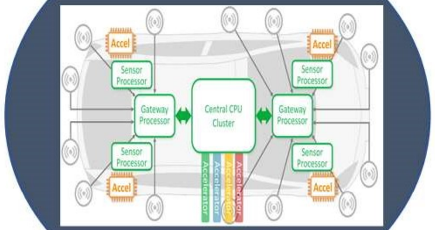
MOTIVATION



Why do we need Predictable, Secure, and Verified CPS?

AIRBAG CONTROL UNIT

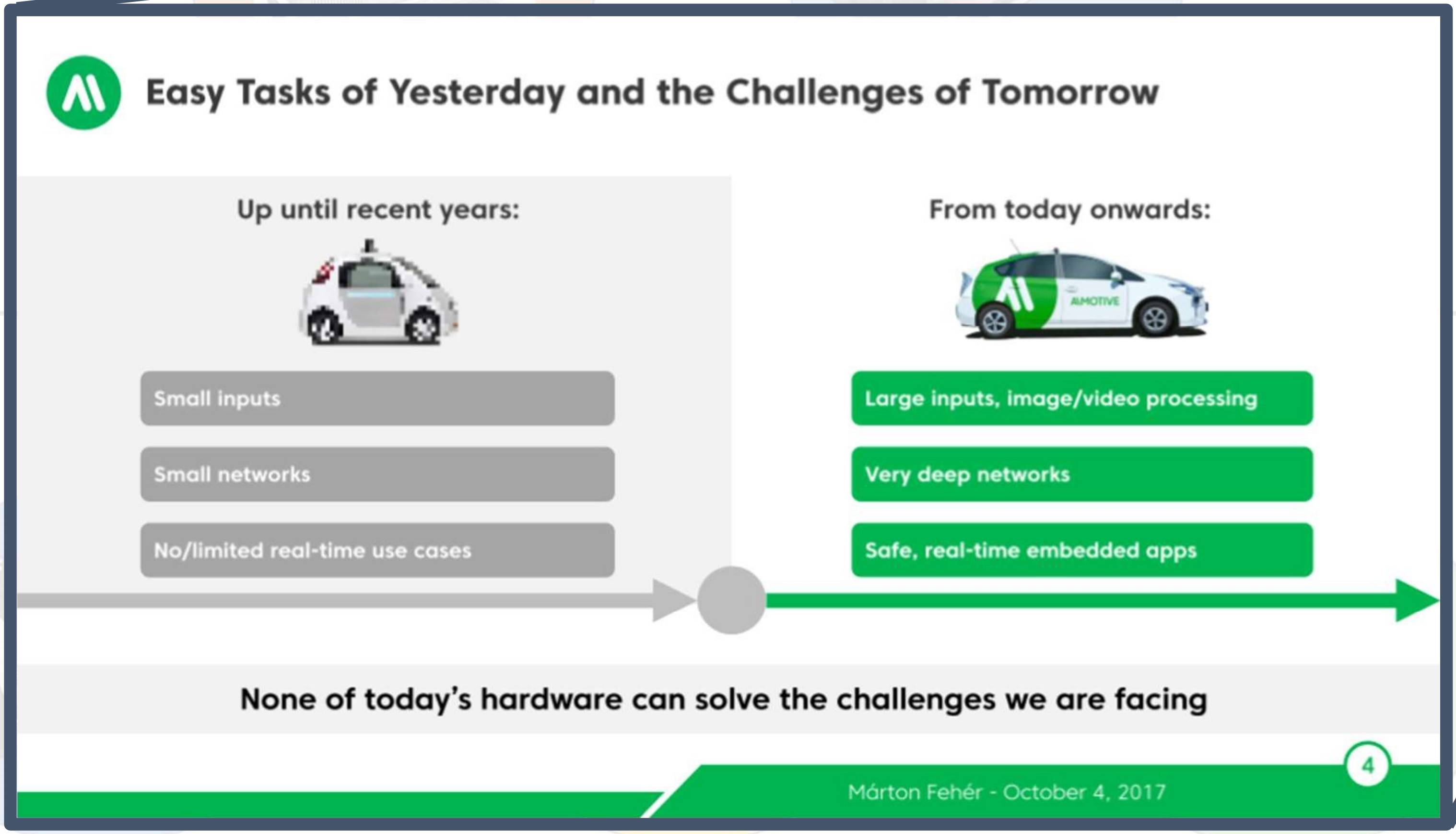




Automotive

Now Near

IoT/Smart Homes



Easy Tasks of Yesterday and the Challenges of Tomorrow

Up until recent years:

- Small inputs
- Small networks
- No/limited real-time use cases

From today onwards:

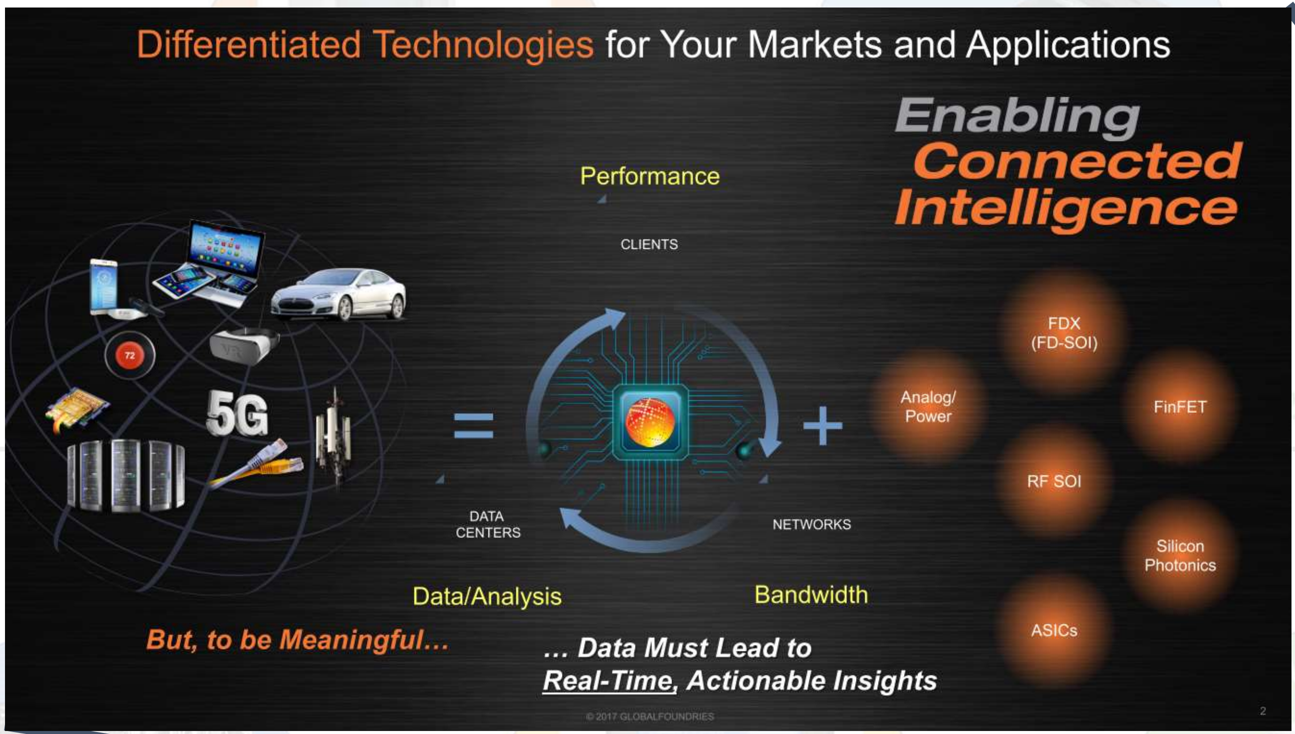
- Large inputs, image/video processing
- Very deep networks
- Safe, real-time embedded apps

None of today's hardware can solve the challenges we are facing

4

Márton Fehér - October 4, 2017

Automotive



Differentiated Technologies for Your Markets and Applications

Enabling Connected Intelligence

Performance

CLIENTS

DATA CENTERS

NETWORKS

Data/Analysis

Bandwidth

5G

=

+

FDX (FD-SOI)

Analog/Power

FinFET

RF SOI

Silicon Photonics

ASICs

But, to be Meaningful…

… Data Must Lead to Real-Time, Actionable Insights

2

010s

Now-Near

earables

IoT/Smart Homes

Smart Window
Gas leak sensor
Smoke sensor
IR Camera
Socket
Light Switch
Air-conditioner
Wind&Rain Sensor
Camera
Body Sensor
Light Switch
Video door phone system
Door magnet sensor
Wireless IR detector
TV set
Home Sever
SOS button
3.5" Master panel

NEW YORK POST

Baby monitors are terrifyingly easy to hack, new study says

By Jane Ridley

BBC

Child safety smartwatches 'easy' to hack, watchdog says

By Joseph Venable
Technology reporter

18 October 2017

Some smartwatches designed for children have security
them vulnerable to hackers, a watchdog has warned.

THE VERGE

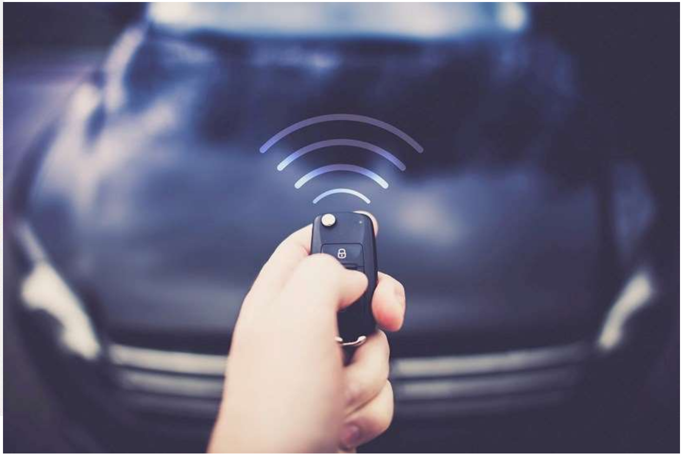
Jeep hackers at it again, this time taking control of steering and braking systems

By Jordan Golson | Aug 2, 2016, 1:45pm EDT

SHARE

25TH USENIX SECURITY SYMPOSIUM

Lock It and Still Lose It —on the (In)Security of Automotive Remote Keyless Entry Systems

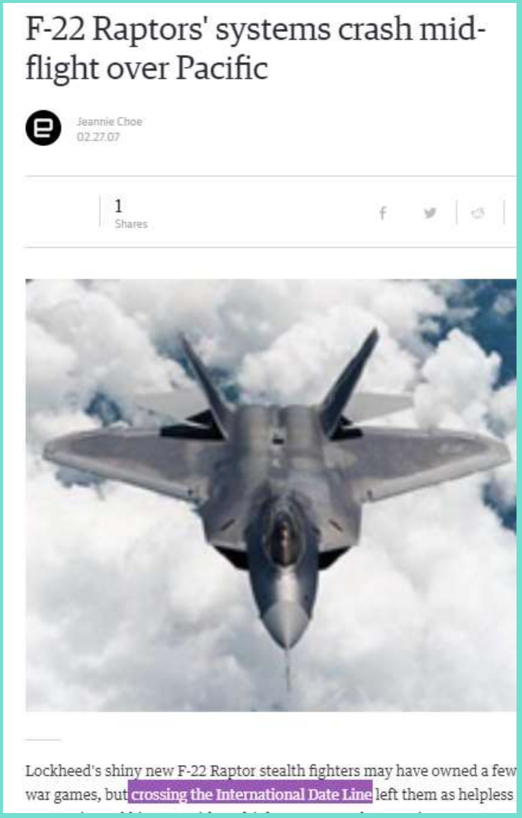


Secure Cyber-Physical Systems

MOTIVATION

Verified Cyber-Physical Systems

MOTIVATION



F-22 Raptors' systems crash mid-flight over Pacific

Jeannie Choe
02.27.07

1
Shares

Lockheed's shiny new F-22 Raptor stealth fighters may have owned a few war games, but crossing the International Date Line left them as helpless

- In 2007, 12 F-22s were going from Hawaii to Japan.
- After crossing the IDL, all 12 experienced multiple crashes.
  - ➢ No navigation
  - ➢ No fuel subsystems
  - ➢ Limited communications
  - ➢ Rebooting didn't help

- Mercedes Class A failed the moose test in 1997.
- Sensors on roof detect overturn and automatically open door.
- What happens if a thief jumps on the car roof?

The New York Times

ARCHIVES | 1997

Mercedes-Benz Tries to Put a Persistent Moose Problem to Rest

By EDMUND L. ANDREWS DEC. 11, 1997

- **Unlike traditional real-time embedded systems:**

Why do we need CPSoCs?

~ 20 million lines of code in S Class Mercedes-Benz

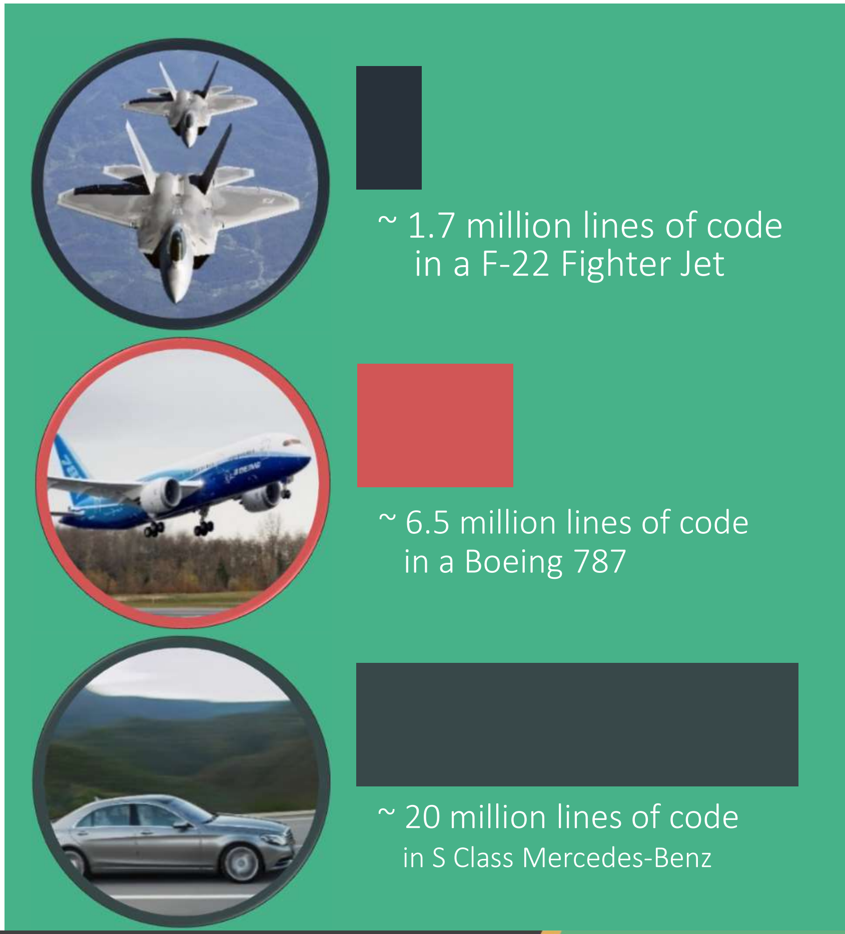
Challenge 1: Computation and Data intensive CPS

MOTIVATION

Challenge 1: Computation and Data intensive CPS

MOTIVATION

- **Unlike traditional real-time embedded systems:**
  - Advanced CPS require significant computational power

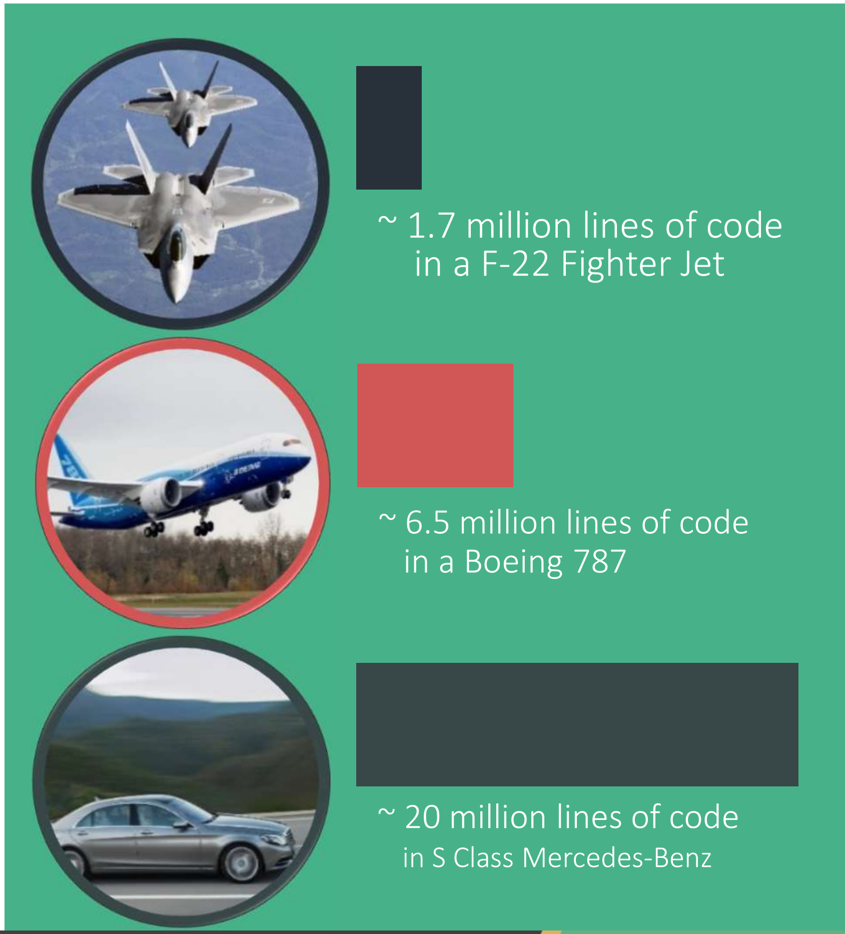


~ 1.7 million lines of code in a F-22 Fighter Jet

~ 6.5 million lines of code in a Boeing 787

~ 20 million lines of code in S Class Mercedes-Benz

- **Unlike traditional real-time embedded systems:**
  - Advanced CPS require significant computational power



~ 1.7 million lines of code in a F-22 Fighter Jet

~ 6.5 million lines of code in a Boeing 787

~ 20 million lines of code in S Class Mercedes-Benz

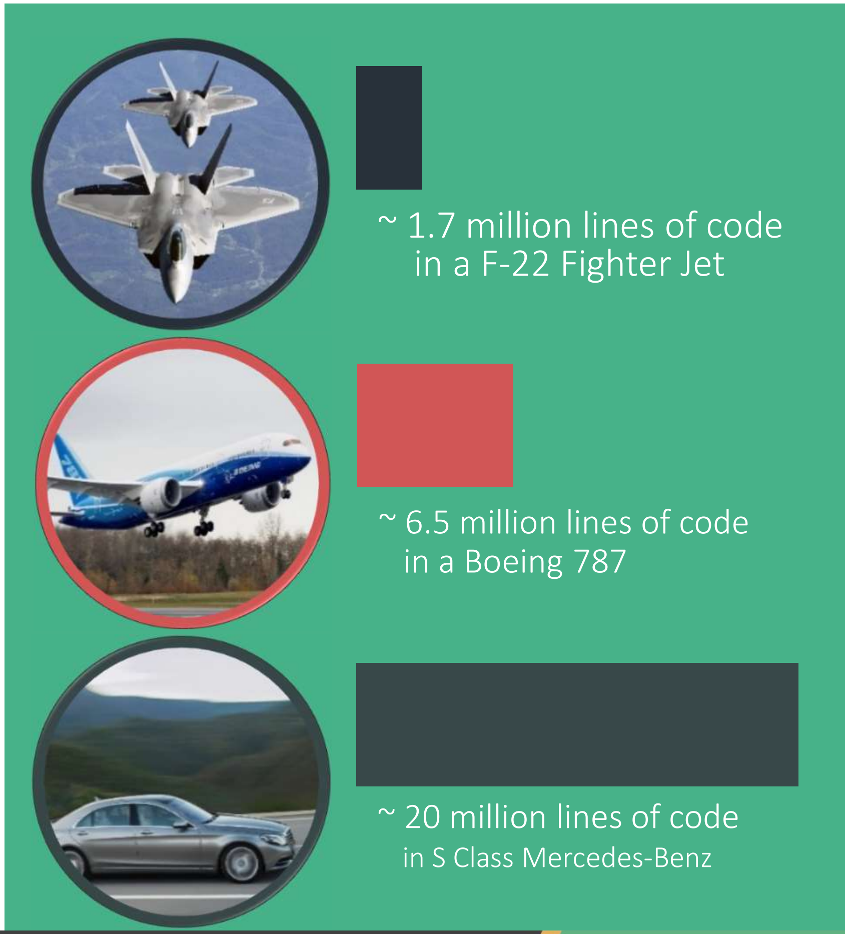
Challenge 1: Computation and Data intensive CPS

MOTIVATION

Challenge 1: Computation and Data intensive CPS

MOTIVATION

- **Unlike traditional real-time embedded systems:**
  - Advanced CPS require significant computational power
  - Autonomous vehicles deploy complex sensor processing and sensor fusion capabilities which are both computation and data intensive



~ 1.7 million lines of code in a F-22 Fighter Jet

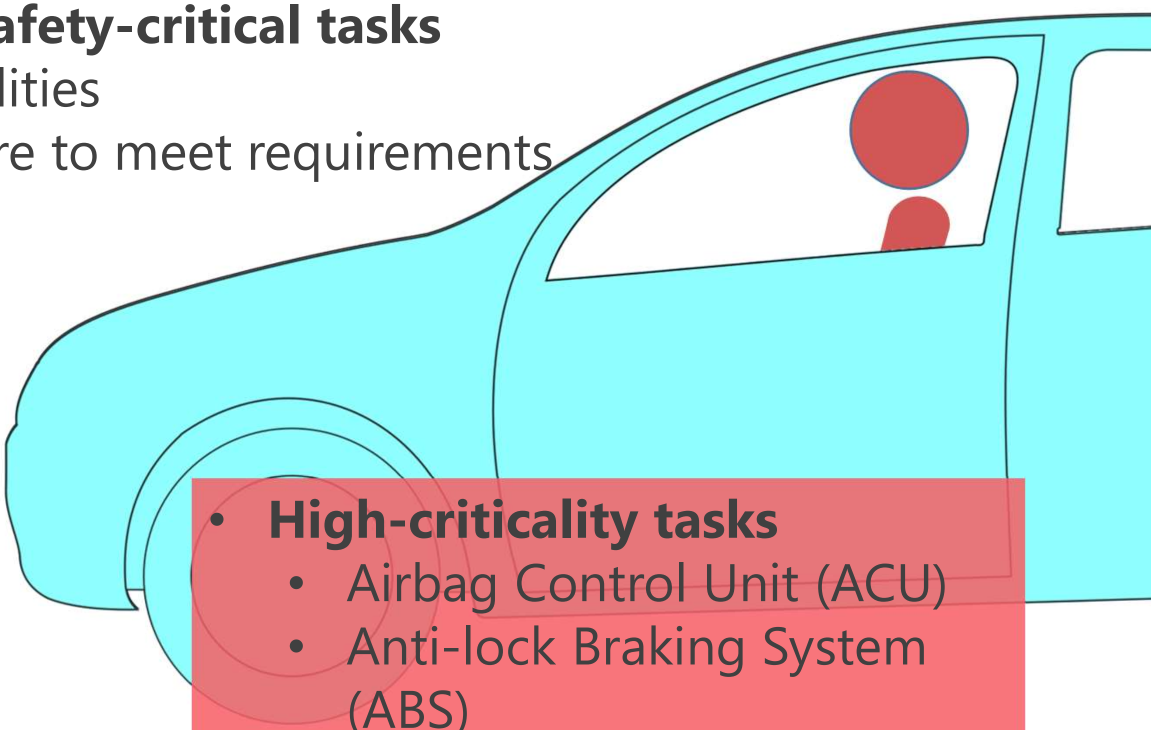
~ 6.5 million lines of code in a Boeing 787

~ 20 million lines of code in S Class Mercedes-Benz

Challenge 2:Mixed-Criticality Nature of CPS

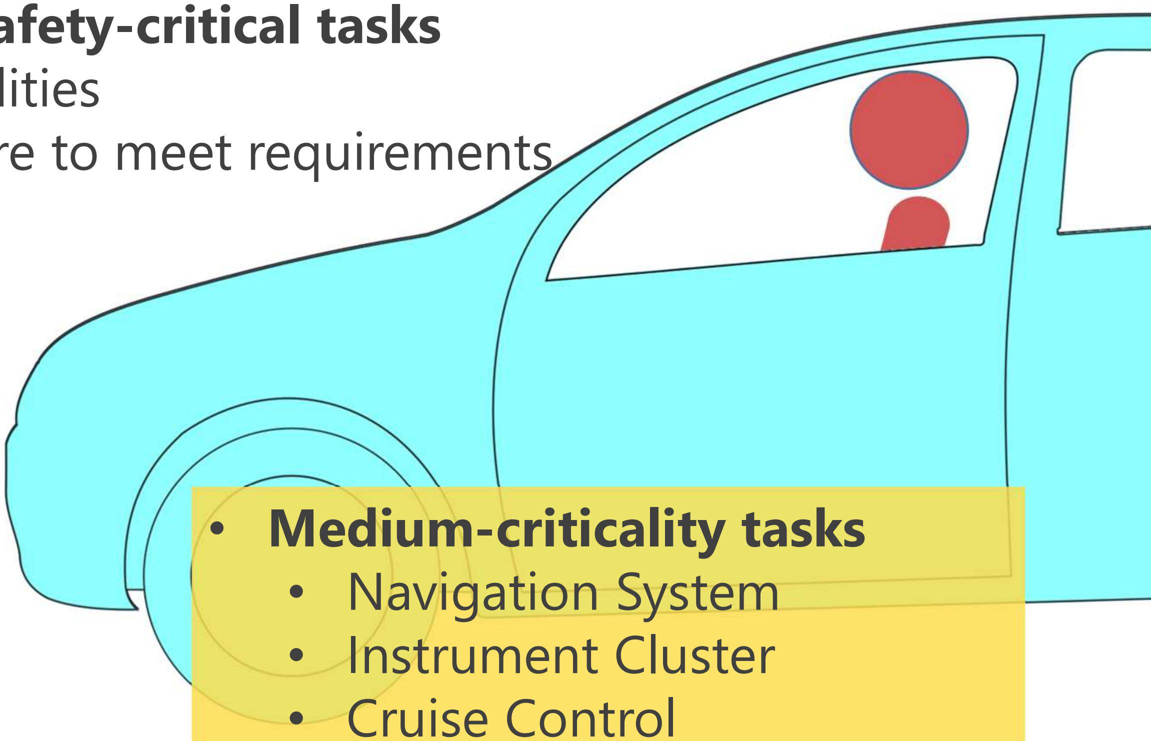
MOTIVATION

- **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)

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

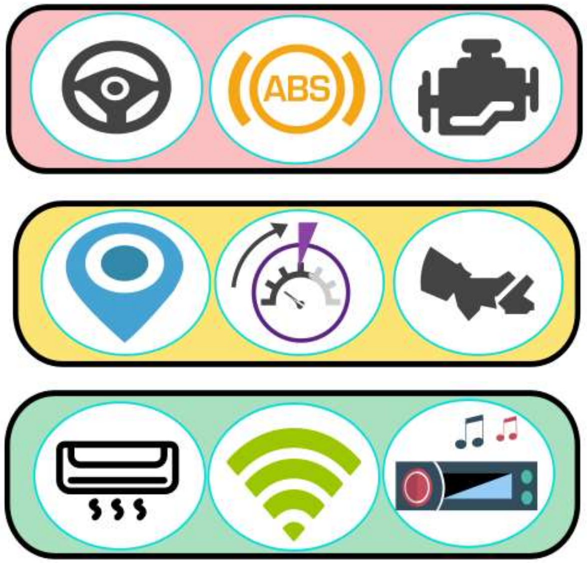


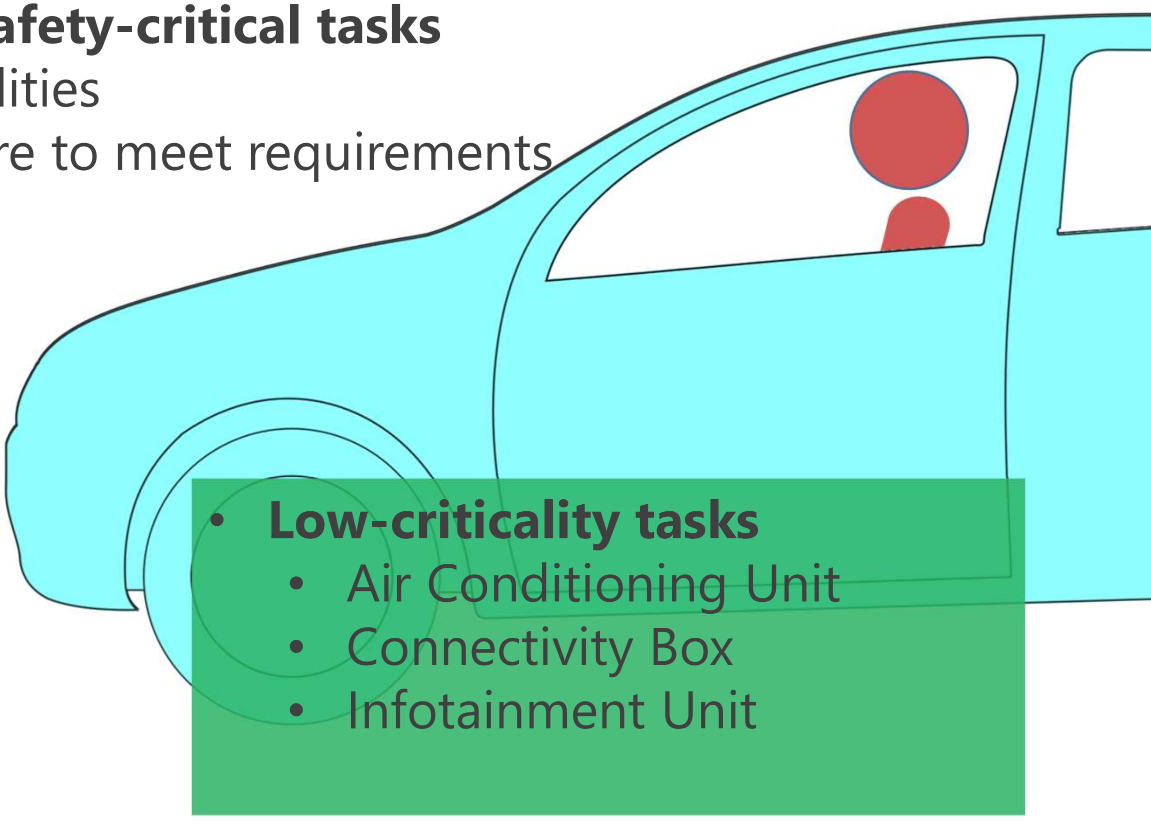
- **Medium-criticality tasks**
  - Navigation System
  - Instrument Cluster
  - Cruise Control

Challenge 2:Mixed-Criticality Nature of CPS

MOTIVATION

- **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

Challenge 2:Mixed-Criticality Nature of CPS

MOTIVATION

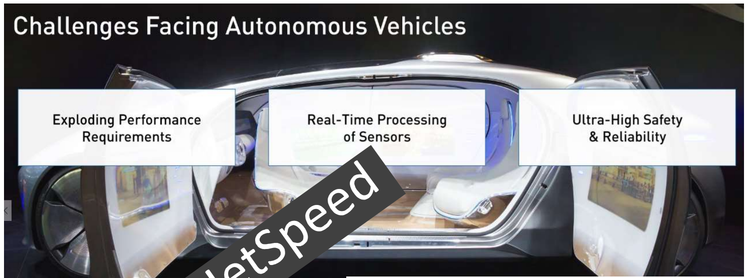
Mixed Criticality Systems

arm

Enabling A Mixed Safety-Criticality Future with Cortex-A76AE

Govind Wathan, Senior Product Manager
Linley Fall Processor Conference
1^{st} November 2018

Increased need for performance and mixed criticality as we move from assisted to autonomous driving systems



Challenges Facing Autonomous Vehicles

Exploding Performance Requirements

Real-Time Processing of Sensors

Ultra-High Safety & Reliability

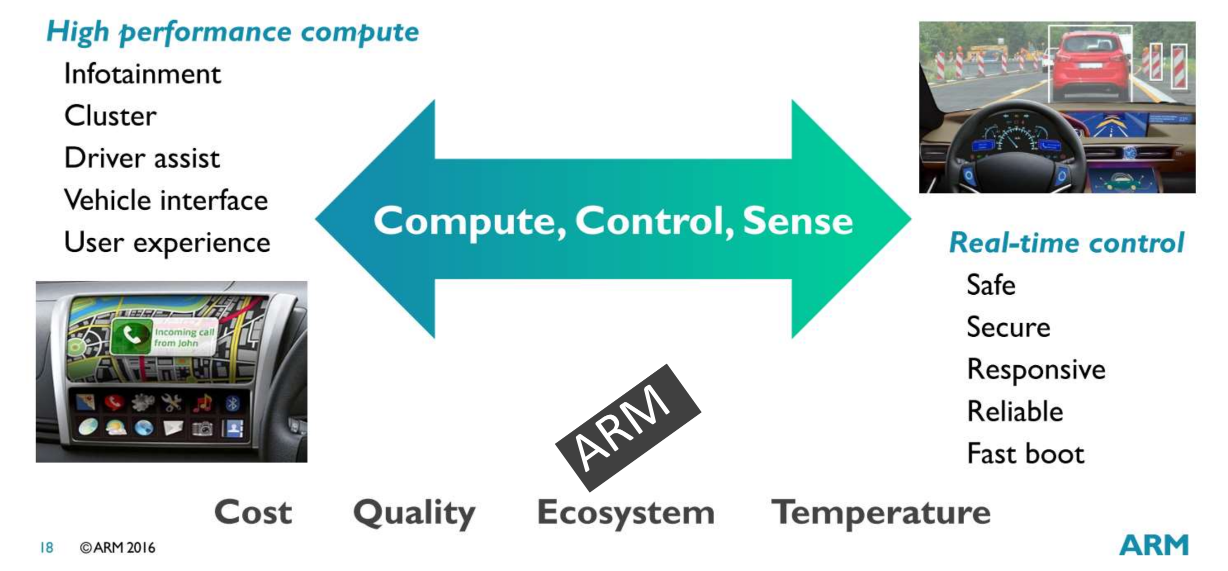
Key Requirements of Automotive-Grade IP

Reduce Risk and Accelerate Qualification for Automotive SoCs

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



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

18 ©ARM 2016

Mixed Criticality Systems

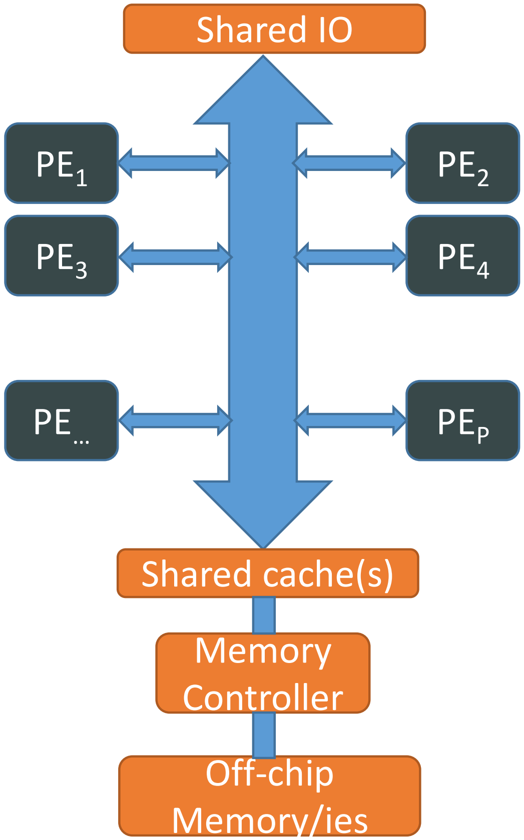
MOTIVATION

Solution to these challenges:

Multiple Processor Systems-on-Chip (MPSoCs)

MPSoCs

MOTIVATION

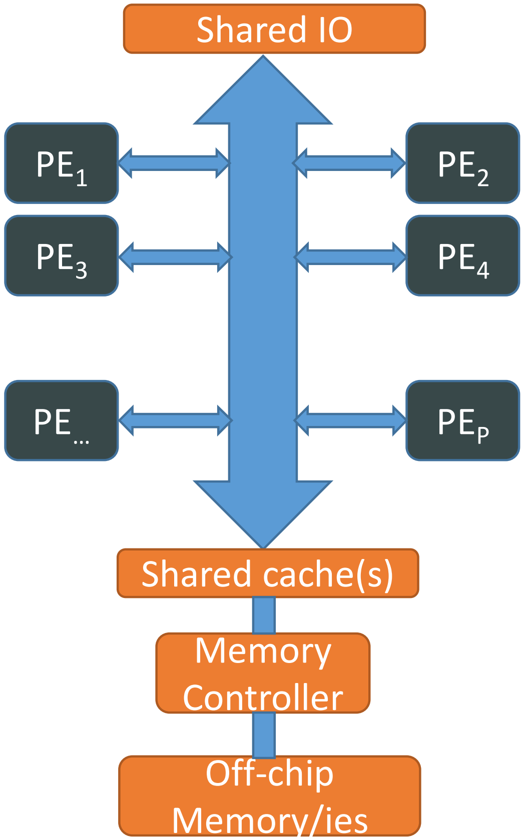


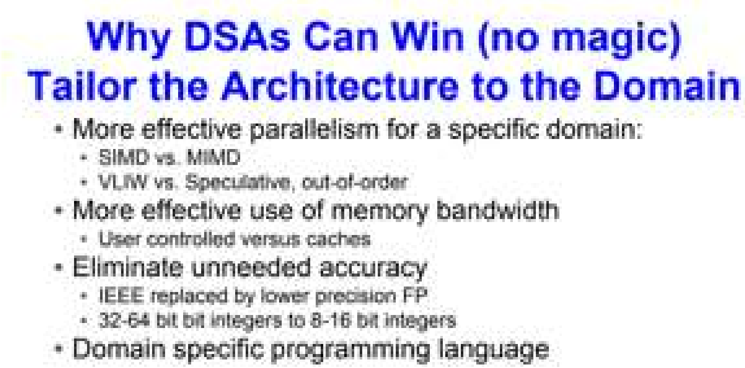
Why MPSoCs?

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

MPSoCs

MOTIVATION





Why DSAs Can Win (no magic)
Tailor the Architecture to the Domain

- More effective parallelism for a specific domain:
  - SIMD vs. MIMD
  - VLIW vs. Speculative, out-of-order
- More effective use of memory bandwidth
  - User controlled versus caches
- Eliminate unneeded accuracy
  - IEEE replaced by lower precision FP
  - 32-64 bit bit integers to 8-16 bit integers
- Domain specific programming language

Hennessy & Patterson, Turing Lecture,
A New Golden Age for Computer Architecture

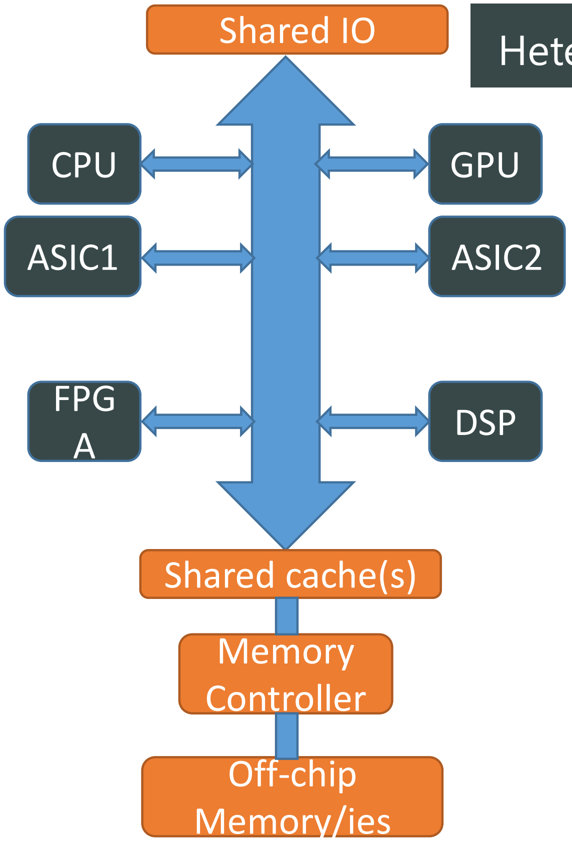
MPSoCs

MOTIVATION

Why Heterogenous MPSoCs?

- Variety of processing capabilities

→ Best-suits MCS conflicting requirements



Heterogenous MPSoCs

Heterogenous MPSoCs

MOTIVATION

Complementary SoC processor requirements

High performance compute

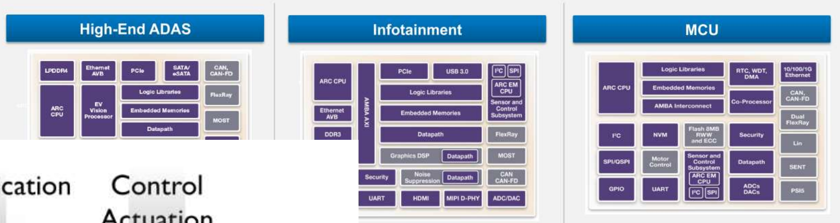
- Infotainment
- Cluster
- Driver assist
- Vehicle interface
- User experience

Real-time control

Cost **Quality** **Ecosystem**

18 ©ARM 2016

Automotive Applications Require Different SoC Architectures



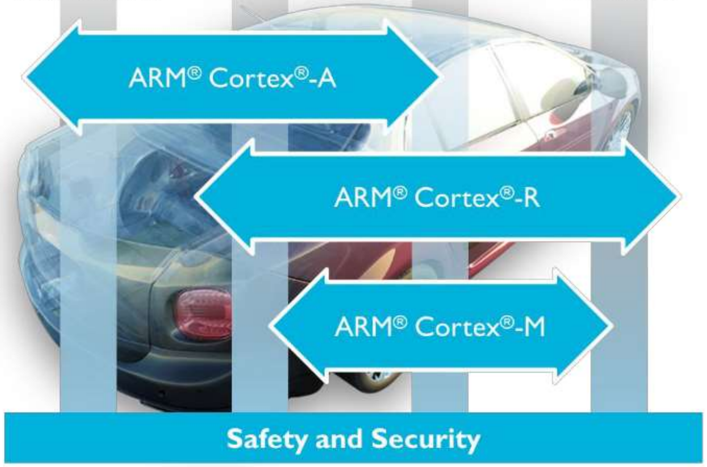
)R4, Ethernet AVB, MIPI, HDMI,
\, ADC, UFS, eMMC
Aultimedia

ion

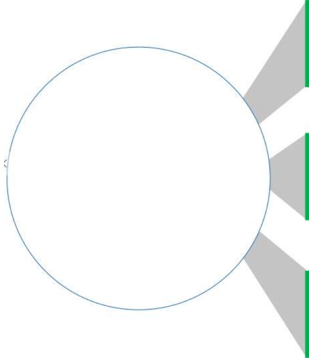
- IP: Ethernet 10/100/1000, ADC, I/F peripherals
- Medium Density NVM

SYNOPSYS

Computation Automation | Sensing | Communication | Control Actuation



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

Linley Autonomous HW Conference 2017 | © Copyright 2017 NetSpeed System

ARM

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

omputing

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

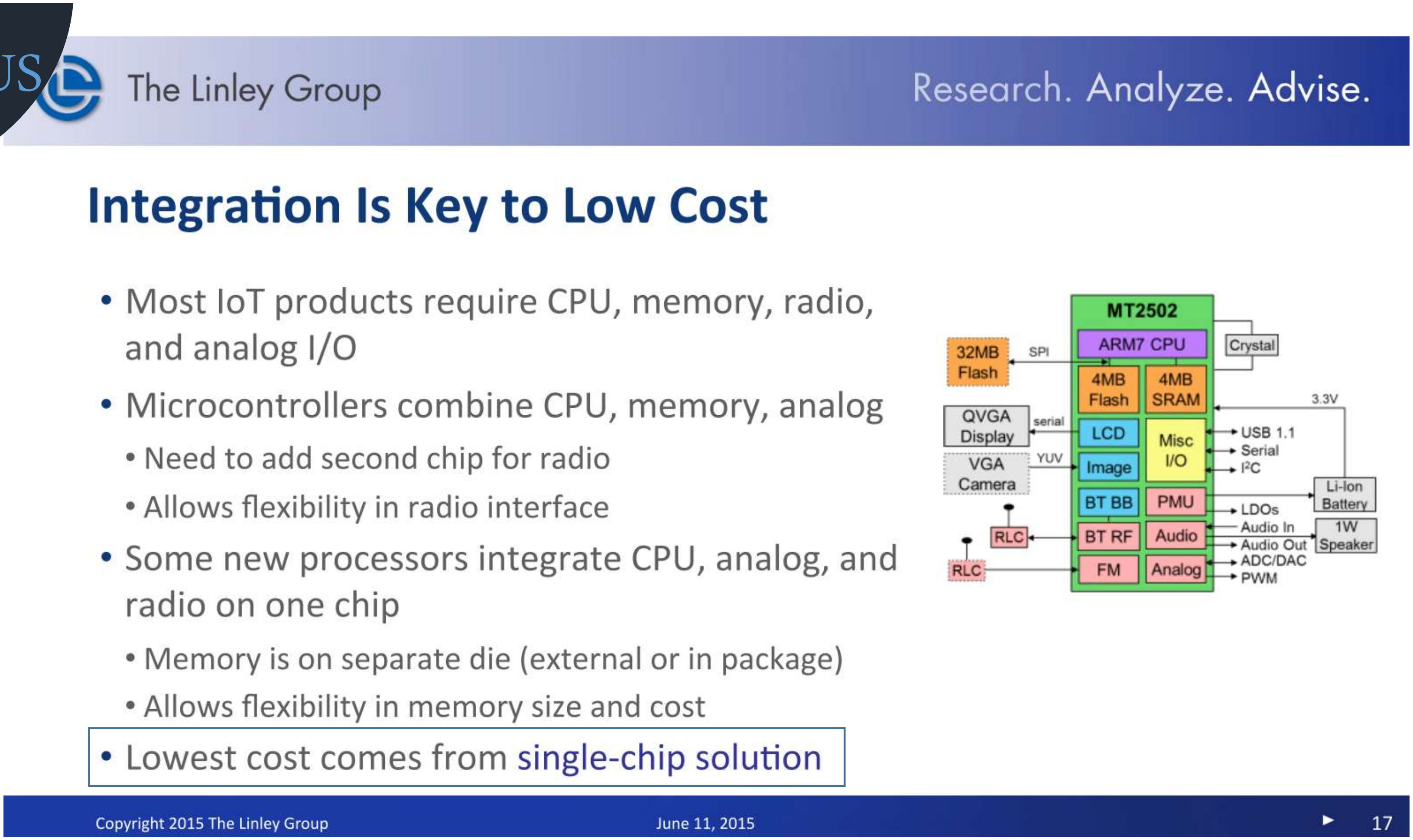
- Lots of data
- Simple computation/item
- Massive parallelism

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

Source: Extreme Te[illegible]gle, ARM

Linley Autonomous HW Conference 2017 | © Copyright 2017 NetSpeed Systems | 5



The Linley Group

Research. Analyze. Advise.

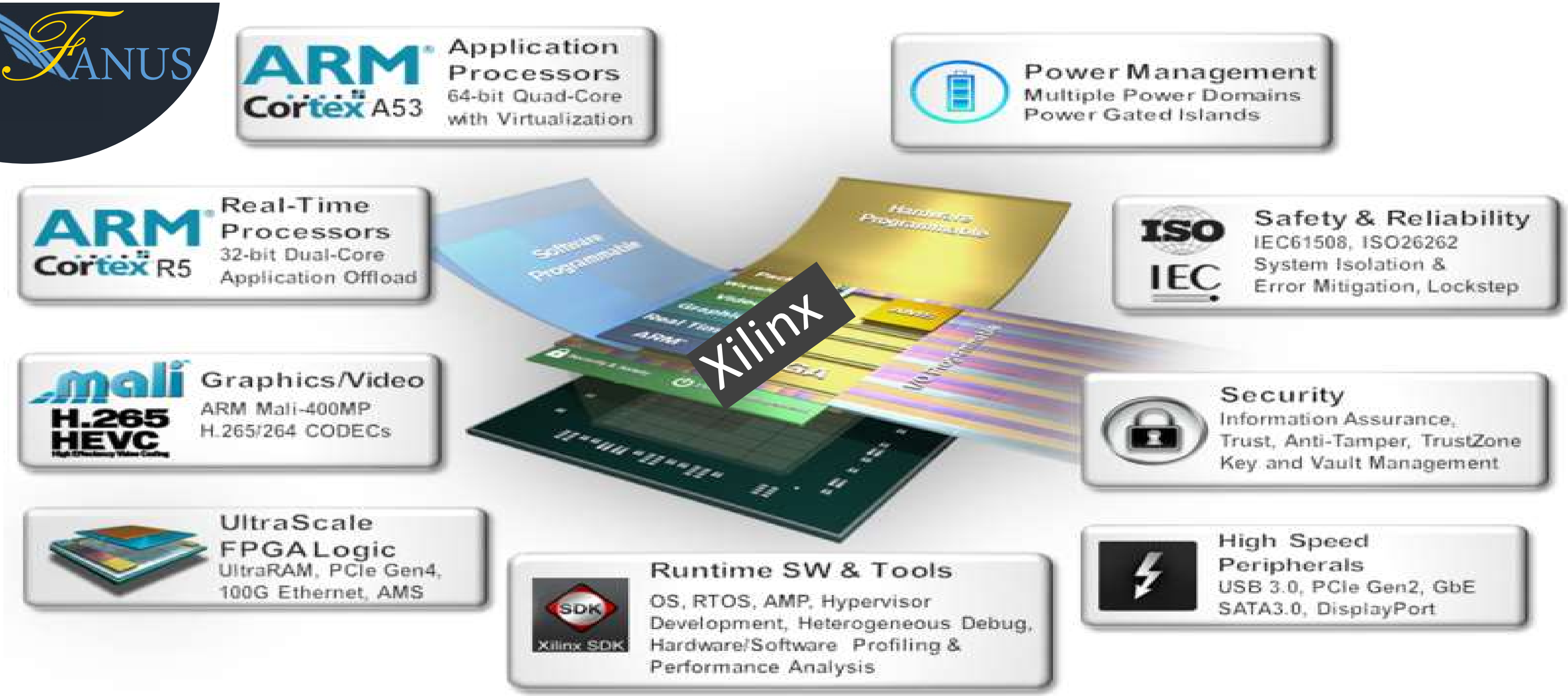
Integration Is Key to Low Cost

- Most IoT products require CPU, memory, radio, and analog I/O
- Microcontrollers combine CPU, memory, analog
  - Need to add second chip for radio
  - Allows flexibility in radio interface
- Some new processors integrate CPU, analog, and radio on one chip
  - Memory is on separate die (external or in package)
  - Allows flexibility in memory size and cost
- Lowest cost comes from single-chip solution

June 11, 2015

17

What about IoT?



ARM Cortex A53 — Application Processors
64-bit Quad-Core
with Virtualization

Power Management
Multiple Power Domains
Power Gated Islands

ARM Cortex R5 — Real-Time Processors
32-bit Dual-Core
Application Offload

ISO IEC — Safety & Reliability
IEC61508, ISO26262
System Isolation &
Error Mitigation, Lockstep

mali H.265 HEVC — Graphics/Video
ARM Mali-400MP
H.265/264 CODECs

Security
Information Assurance,
Trust, Anti-Tamper, TrustZone
Key and Vault Management

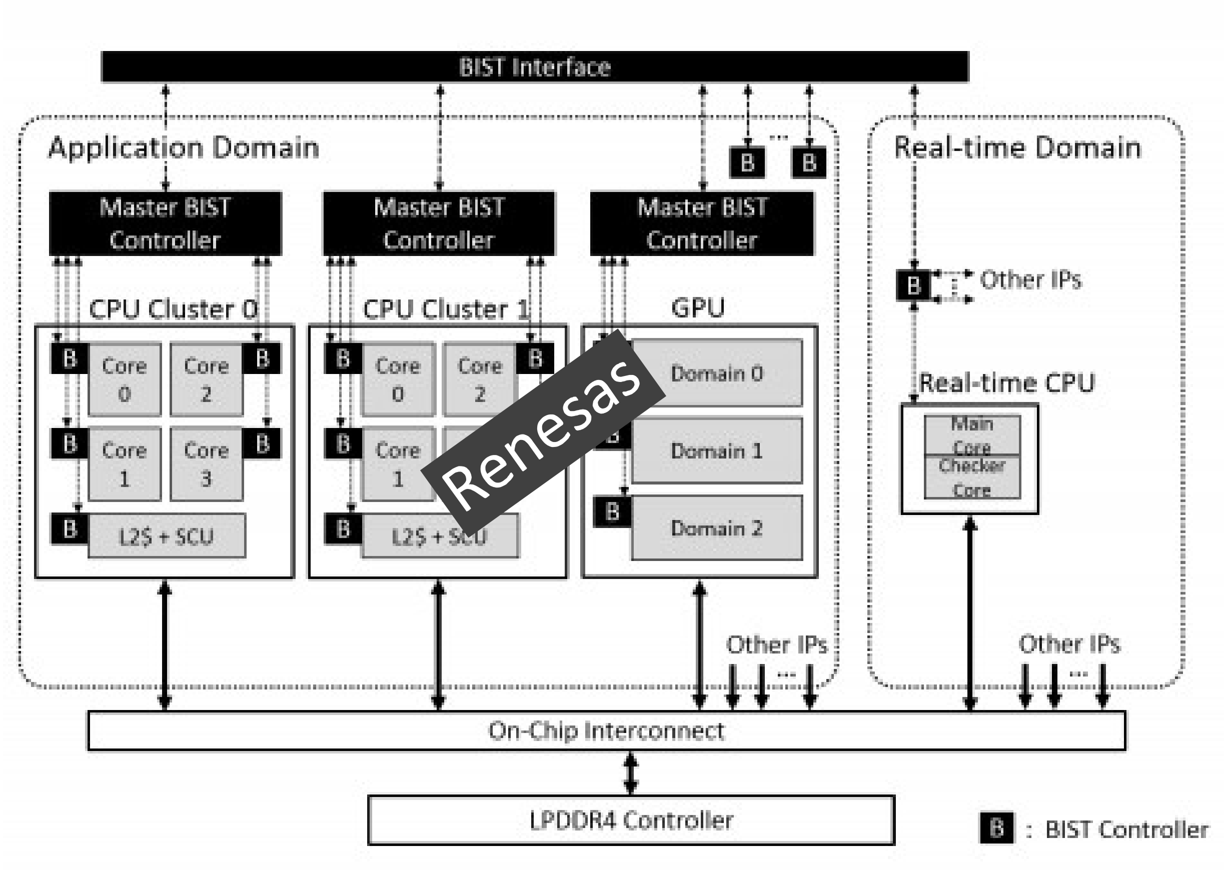
UltraScale FPGA Logic
UltraRAM, PCIe Gen4,
100G Ethernet, AMS

SDK Xilinx SDK — Runtime SW & Tools
OS, RTOS, AMP, Hypervisor
Development, Heterogeneous Debug,
Hardware/Software Profiling &
Performance Analysis

High Speed Peripherals
USB 3.0, PCIe Gen2, GbE
SATA3.0, DisplayPort

Heterogenous MPSoCs with Real-time Processors

MOTIVATION



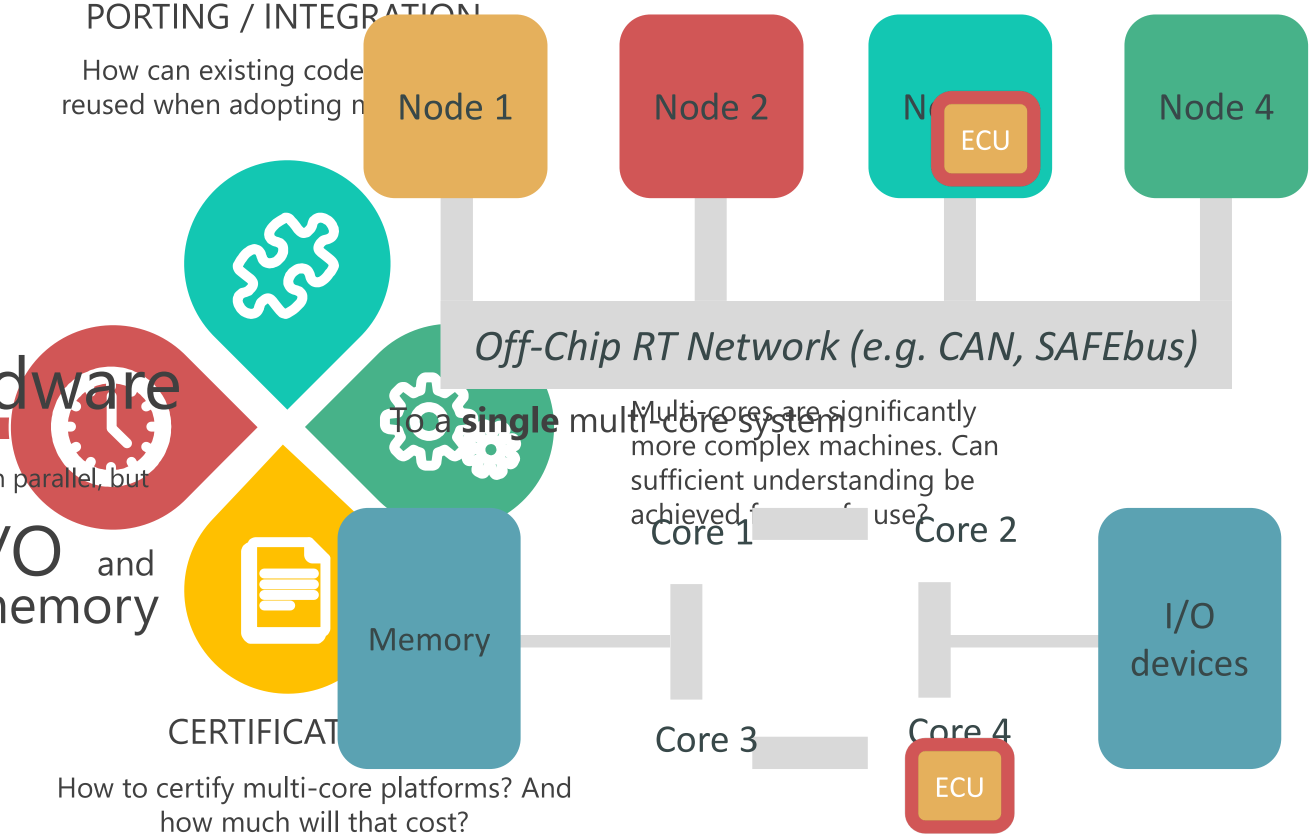
Heterogenous MPSoCs with Real-time Processors

MOTIVATION

From **multiple** single-core systems

PORTING / INTEGRATION

How can existing code
reused when adopting m



To a **single** multi-core system

Multi-cores are significantly more complex machines. Can sufficient understanding be achieved f   f   use?

Sharing Hardware

Sharing Hardware

How to achieve a level of predictability that is equivalent to single-cores without excessive pessimism?

computation is performed in parallel, but

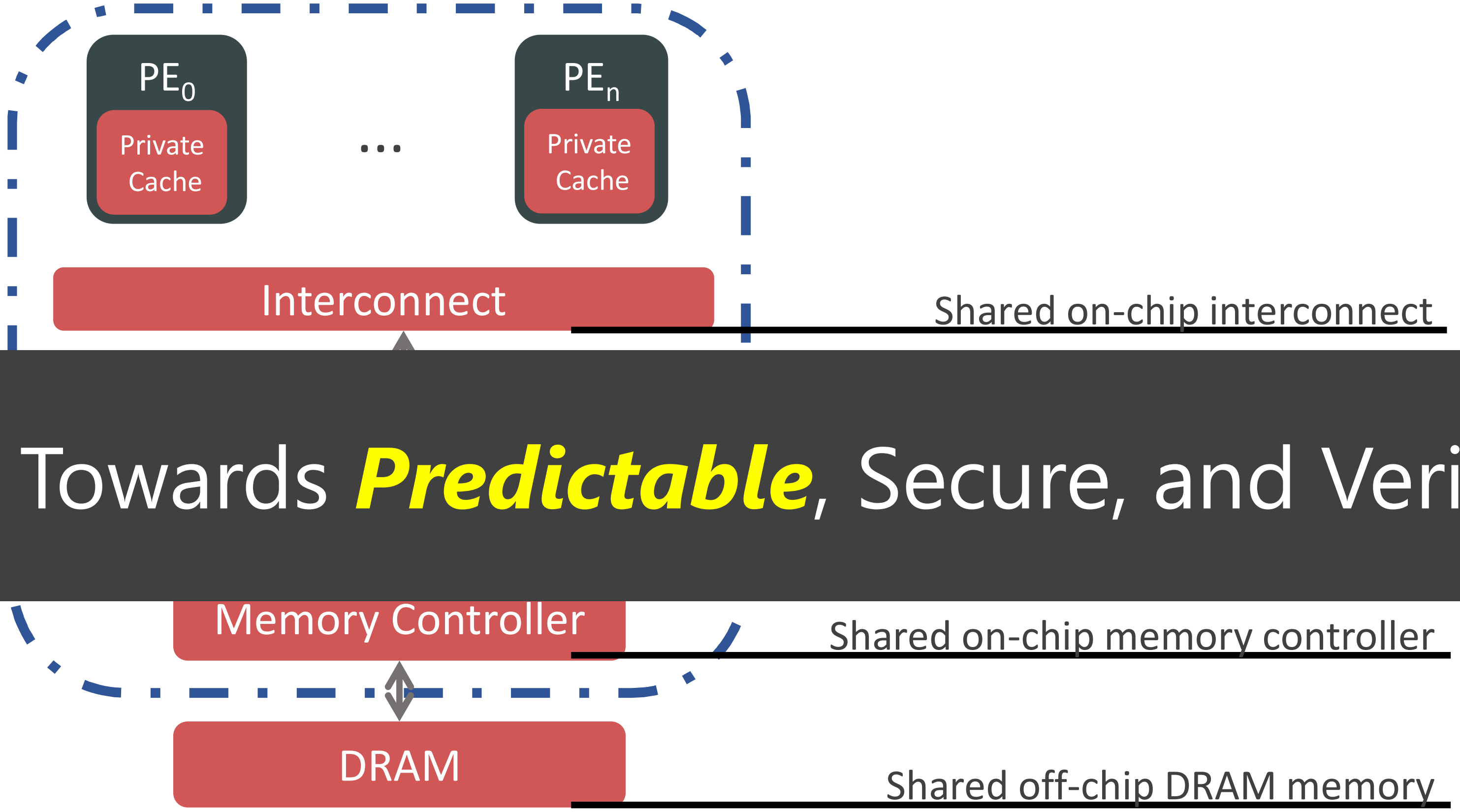
shared I/O and memory

CERTIFICAT

How to certify multi-core platforms? And how much will that cost?

MPSoC Challenges

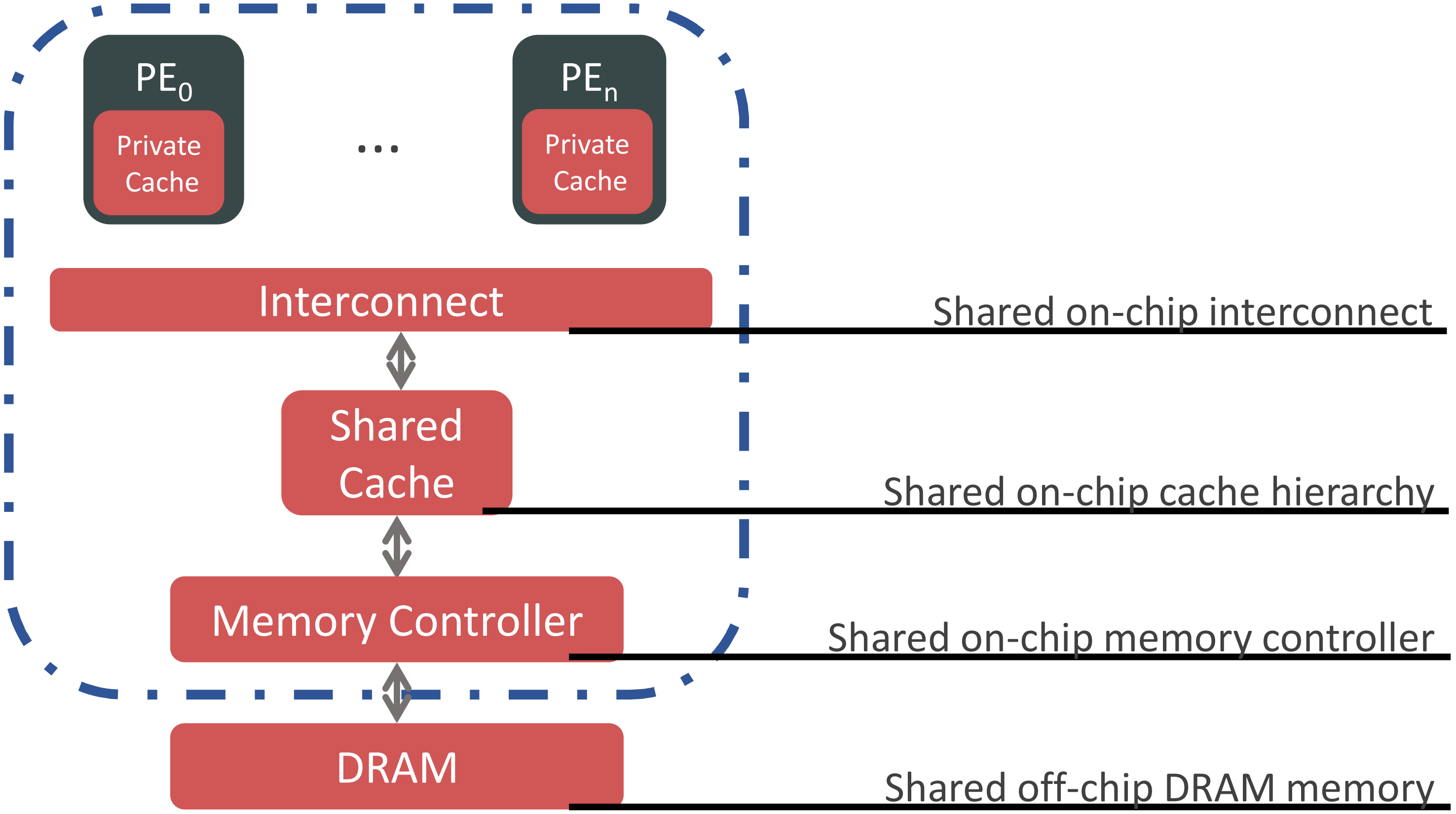
MOTIVATION


PE_0
Private Cache
…
PE_n
Private Cache
Interconnect
Memory Controller
DRAM
Shared on-chip interconnect
Shared on-chip memory controller
Shared off-chip DRAM memory

Towards ***Predictable***, Secure, and Verified CPSoCs

Predictable CPSoC

PREDICTABILITY



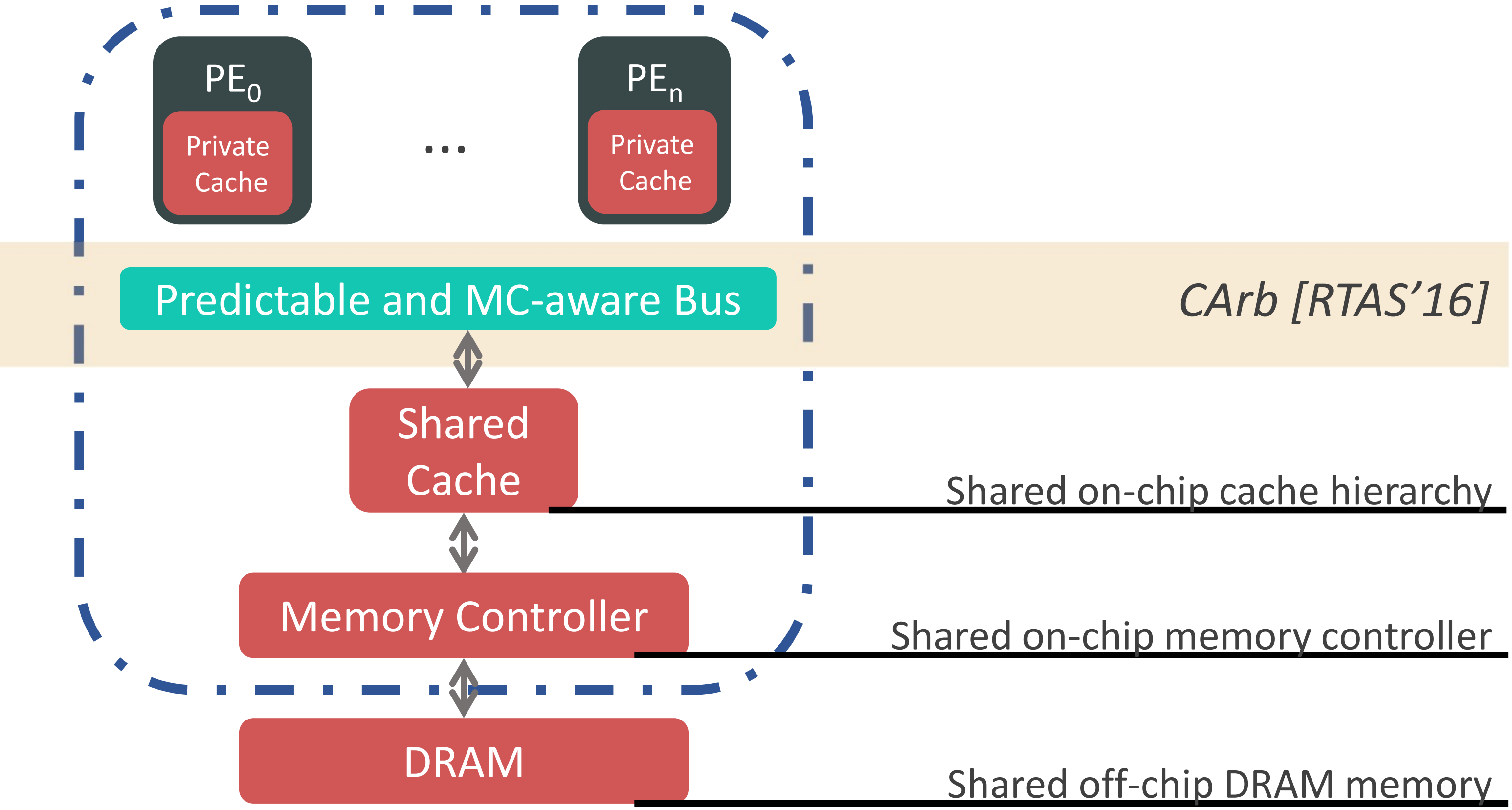
Shared on-chip interconnect

Shared on-chip cache hierarchy

Shared on-chip memory controller

Shared off-chip DRAM memory

Predictable CPSoC



CArb [RTAS'16]

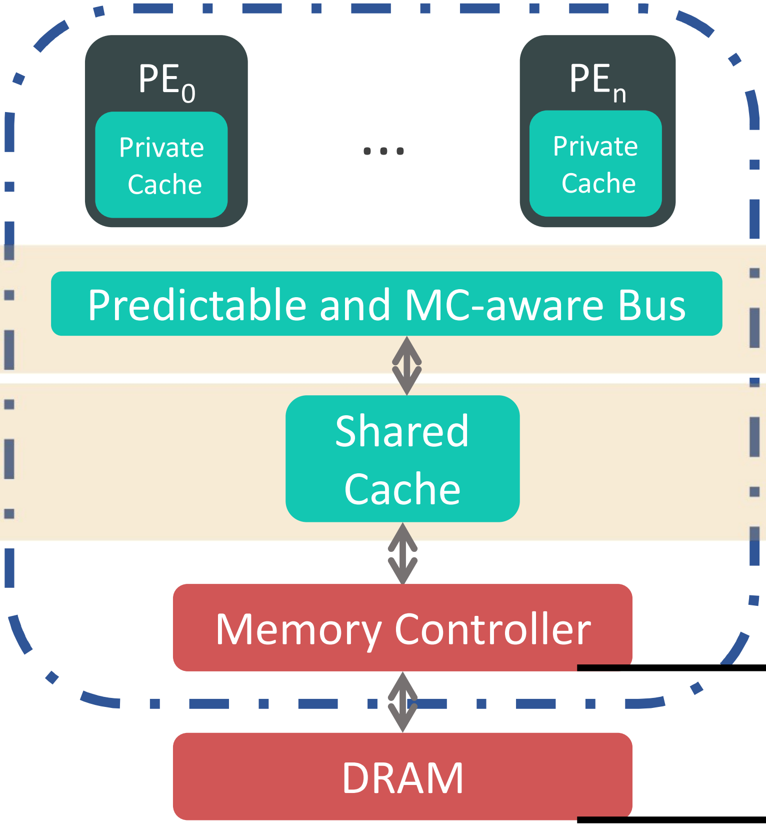
Shared on-chip cache hierarchy

Shared on-chip memory controller

Shared off-chip DRAM memory

Predictable CPSoC

PREDICTABILITY



CArb [RTAS'16]

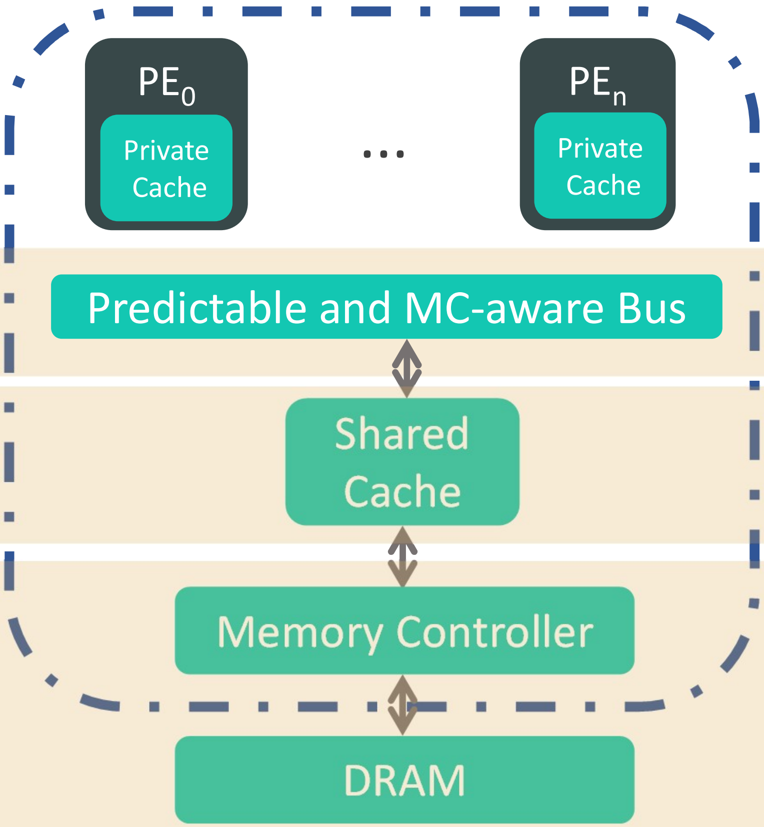
PMSI [RTAS'17], PENDULUM [RTSS'19]
HourGlass [ArXiv'18], Ongoing….

Shared on-chip memory controller

Shared off-chip DRAM memory

Predictable CPSoC

PREDICTABILITY



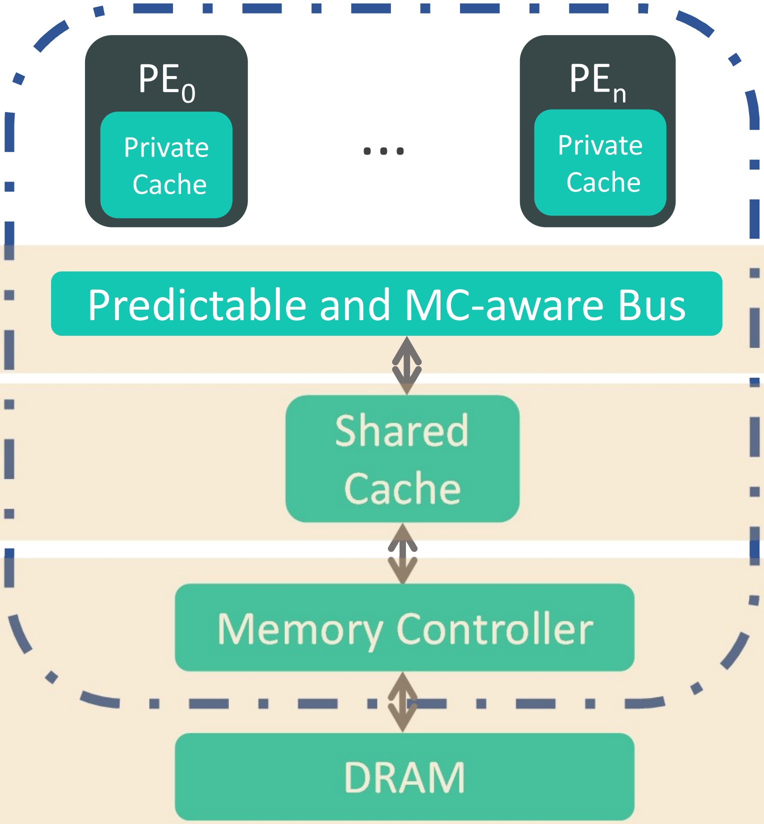
CArb [RTAS'16]

PMSI [RTAS'17], PENDULUM [RTSS'19]
HourGlass [ArXiv'18], Ongoing….

PMC [RTAS'15, TECS'16]
MCSim [TECS'17]
MCS-MPSoCs [EMSOFT'18, TCAD'18]
RLDRAM [RTSS'18]
DRAMbulism [RTAS'20]

Predictable CPSoC

PREDICTABILITY



CArb [RTAS'16]

PMSI [RTAS'17], PENDULUM [RTSS'19]
HourGlass [ArXiv'18], Ongoing….

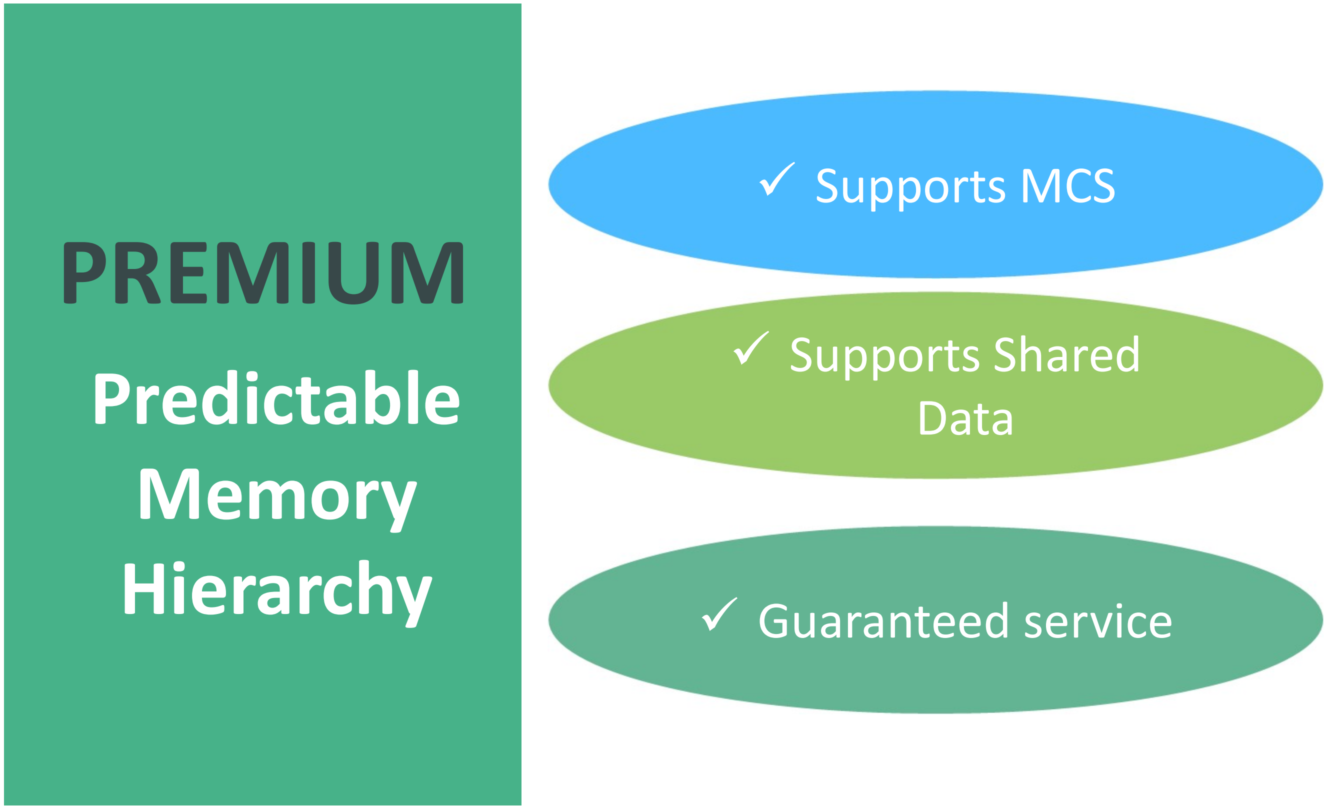
PMC [RTAS'15, TECS'16]
MCSim [TECS'17]
MCS-MPSoCs [EMSOFT'18, TCAD'18]
RLDRAM [RTSS'18]
DRAMbulism [RTAS'20]

PREMIUM

Predictable Memory Hierarchy

Predictable CPSoC

PREDICTABILITY

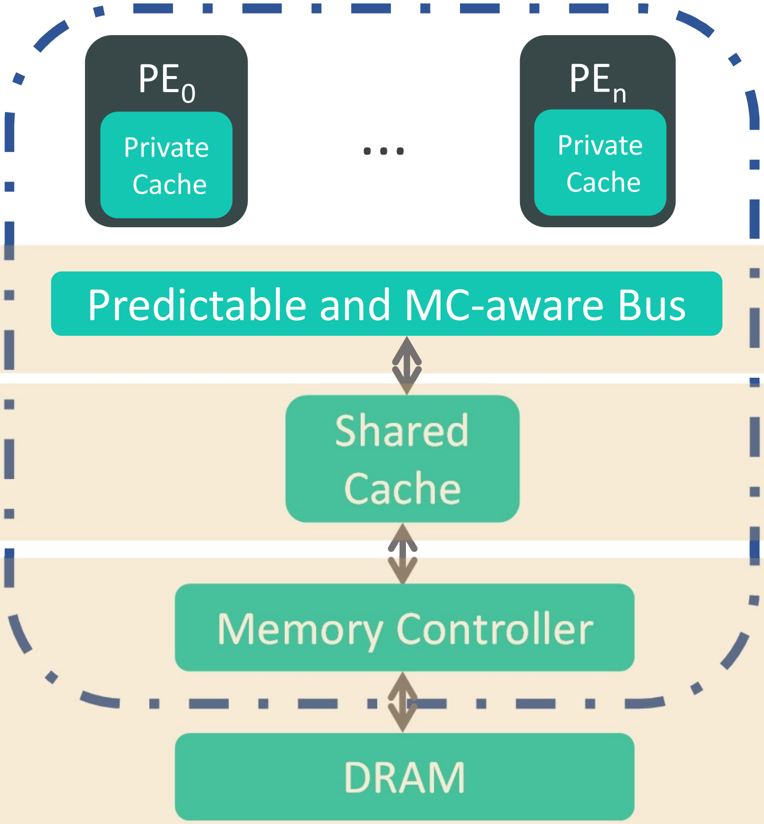


PREMIUM

Predictable Memory Hierarchy

- ✓ Supports MCS
- ✓ Supports Shared Data
- ✓ Guaranteed service

Predictable CPSoC



CArb [RTAS’16]

PMSI [RTAS’17], PENDULUM [RTSS’19]
HourGlass [ArXiv’18], Ongoing….

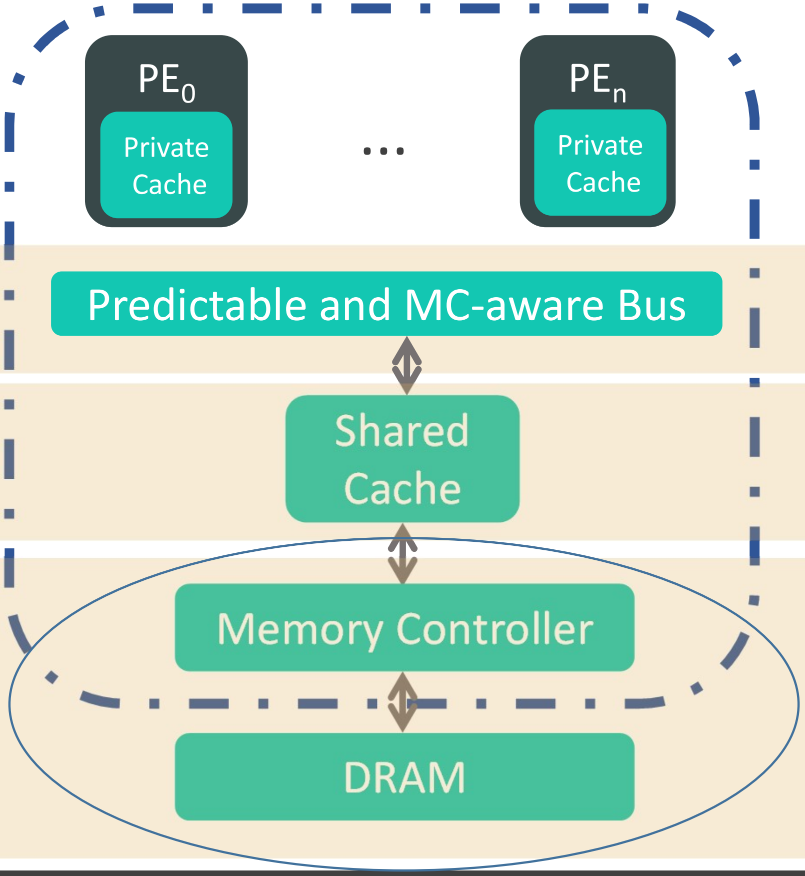
PMC [RTAS’15, TECS’16]
MCSim [TECS’17]
MCS-MPSoCs [EMSOFT’18, TCAD’18]
RLDRAM [RTSS’18]
DRAMbulism [RTAS’20]

PREMIUM

Predictable Memory Hierarchy

Predictable CPSoC

PREDICTABILITY



CArb [RTAS'16]

PMSI [RTAS'17], PENDULUM [RTSS'19]
HourGlass [ArXiv'18], Ongoing….

PMC [RTAS'15, TECS'16]
MCSim [TECS'17]
MCS-MPSoCs [EMSOFT'18, TCAD'18]
RLDRAM [RTSS'18]
DRAMbulism [RTAS'20]

PREMIUM

Predictable Memory Hierarchy

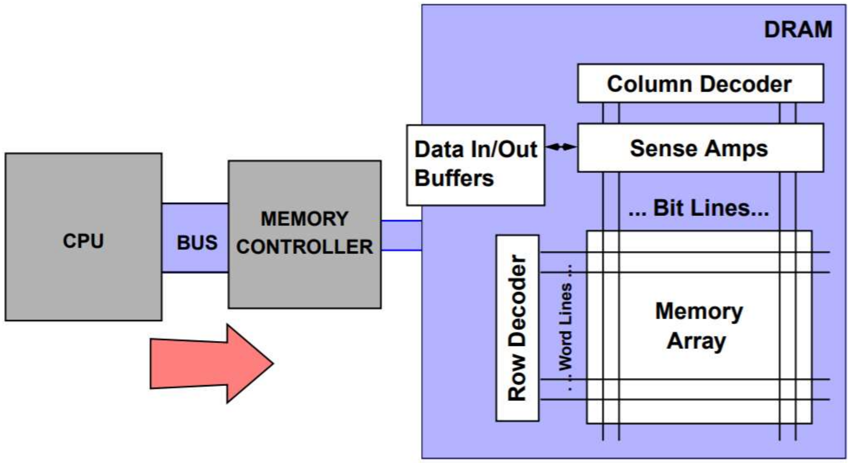
Predictable CPSoC

PREDICTABILITY

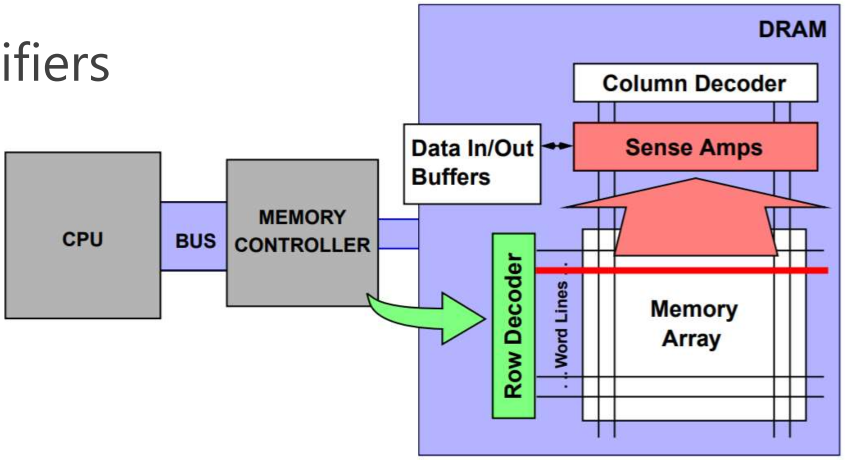
Background

DRAM

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



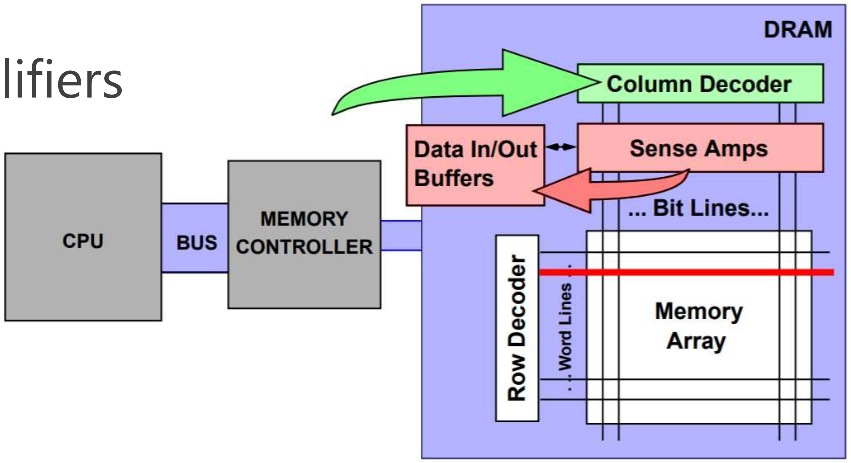
Background

DRAM

Background

DRAM

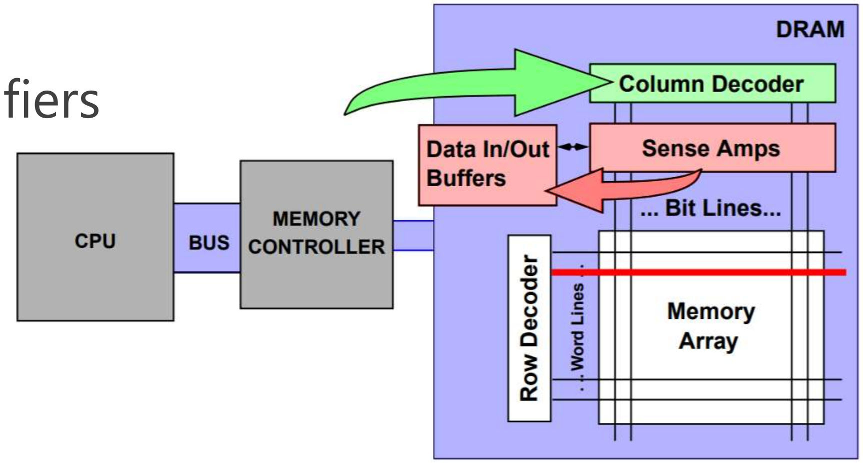
- DRAM Consists of multiple banks
- The memory controller (MC) manages accesses to DRAM
- A request in general consists of:
  - ACTIVATE command:
    - Bring data row from cells into sense amplifiers
  - RD/WR commands:
    - To read/write from specific columns in the sense amplifiers



Background

DRAM

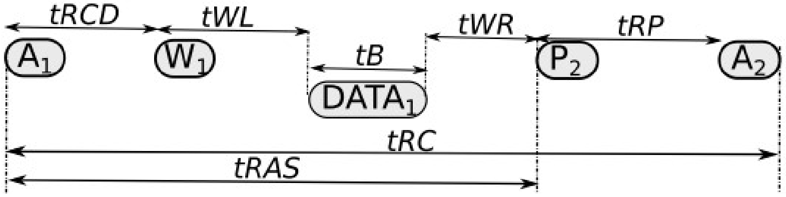
- DRAM Consists of multiple banks
- The memory controller (MC) manages accesses to DRAM
- A request in general consists of:
  - ACTIVATE command:
    - Bring data row from cells into sense amplifiers
  - RD/WR commands:
    - To read/write from specific columns in the sense amplifiers
  - PRECHARGE command: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



Background

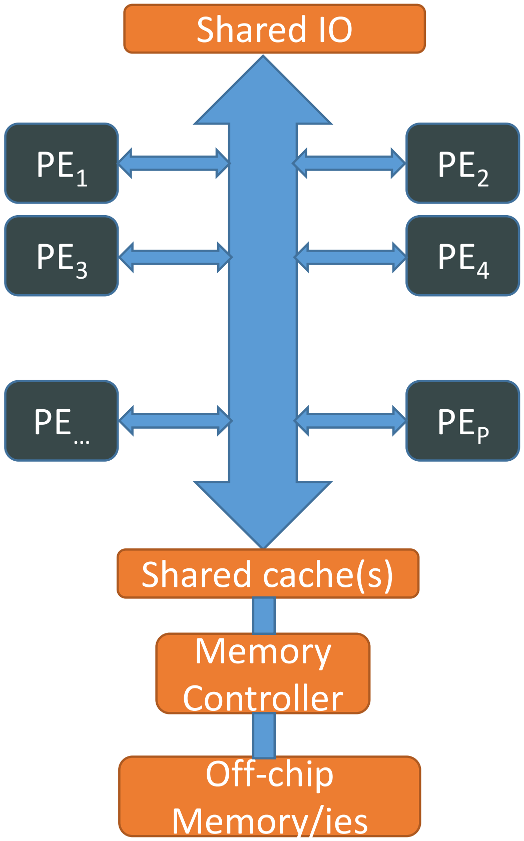
DRAM

- 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



System Overview

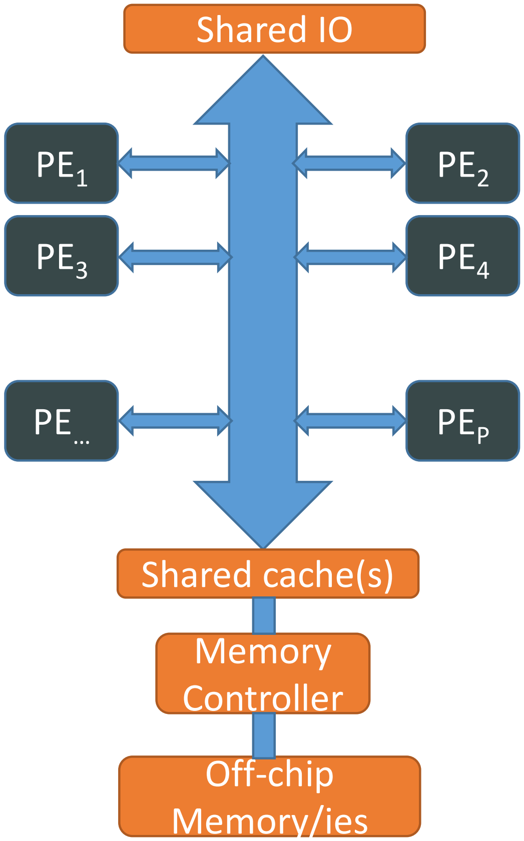
MODEL



- 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

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

Why We Bother?

Challenge: operations of one PE affect the temporal behavior of other PEs, which complicates the timing analysis of the system.

Most of the MCS scheduling techniques do not incorporate these interferences in their scheduling or analysis

Approaches focusing on shared resources mostly assume SMPs

Challenge: operations of one PE affect the

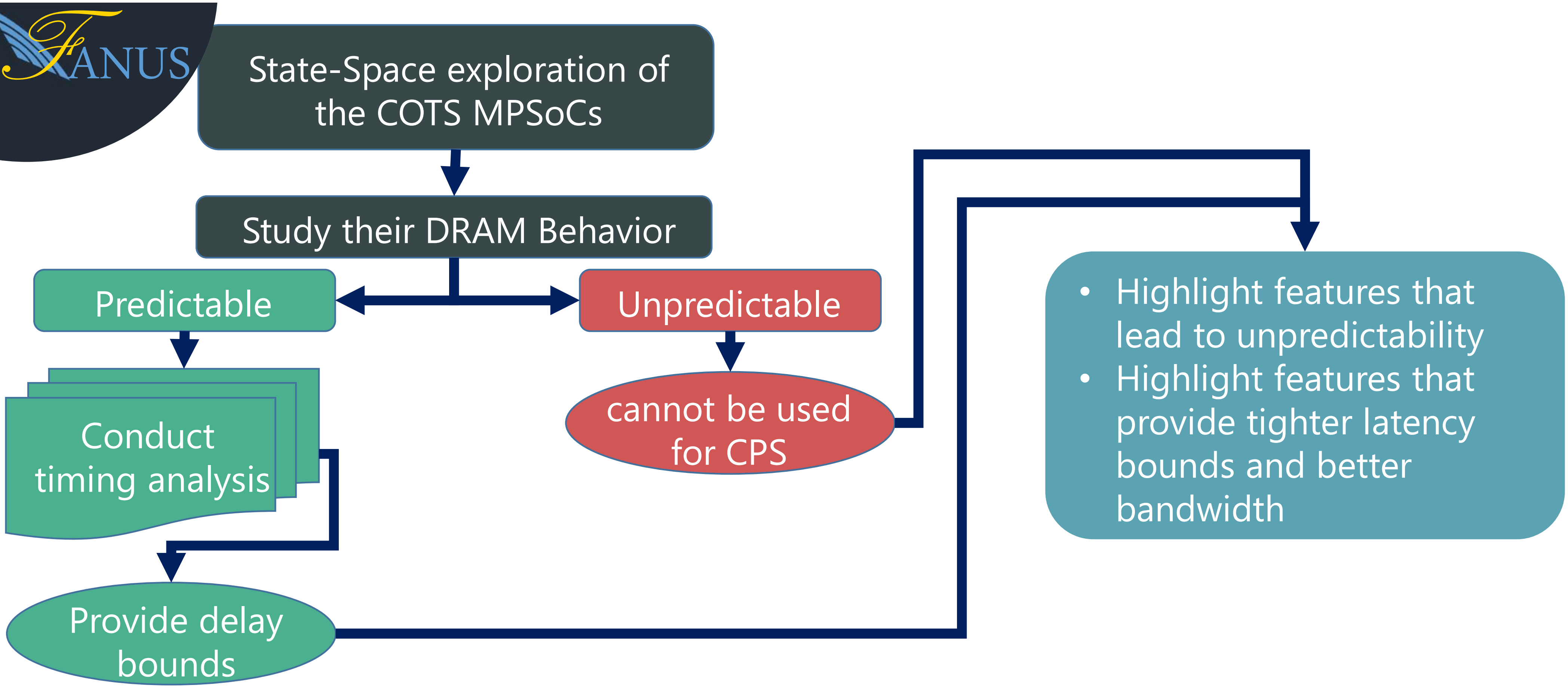
vior of other PEs, which

7.4.2.7 Where the software is to implement both safety and non-safety functions, then all of the software shall be treated as safety-related, unless adequate independence between the functions can be demonstrated in the design.
[IEC61508-3]

tly assume SIMPs

Why We Bother?

MODEL



State-Space exploration of the COTS MPSoCs

Study their DRAM Behavior

Predictable

Unpredictable

Conduct timing analysis

cannot be used for CPS

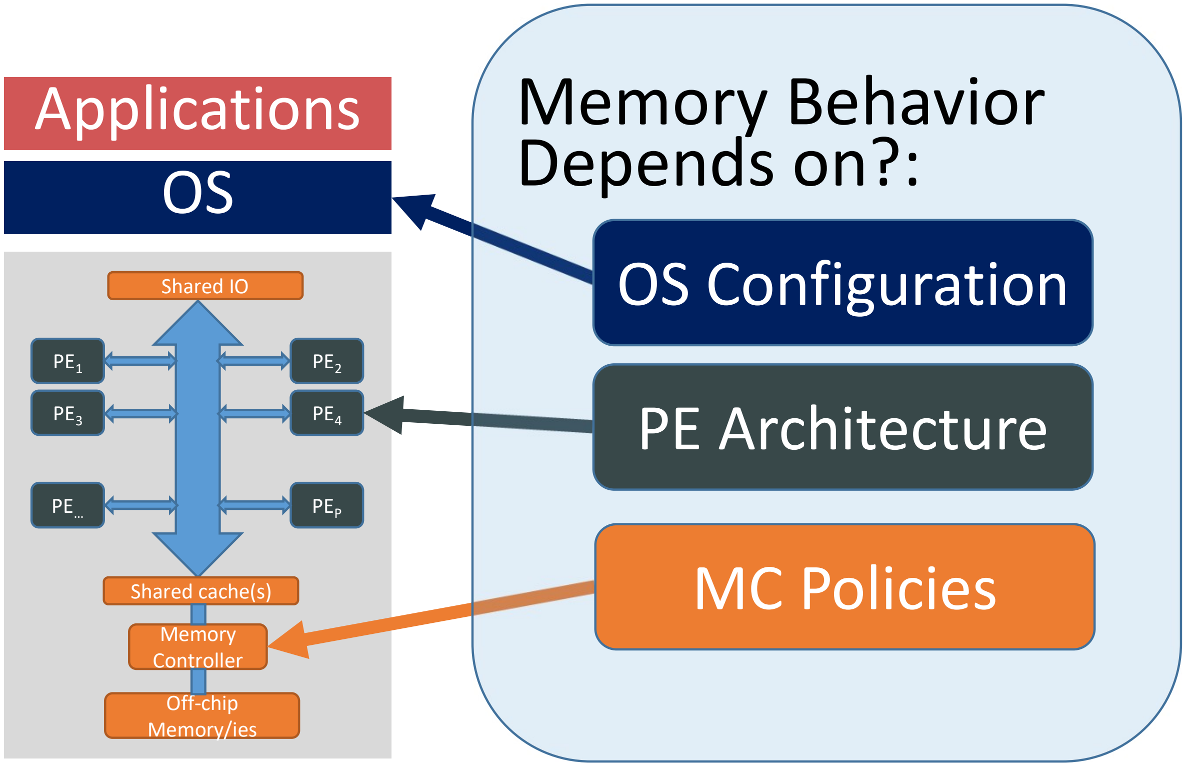
Provide delay bounds

- Highlight features that lead to unpredictability
- Highlight features that provide tighter latency bounds and better bandwidth

Big Picture

*[EMSOFT’18] **Mohamed Hassan**, Rodolfo Pellizzoni, "Bounding DRAM Interference in COTS Heterogeneous MPSoCs for Mixed Criticality Systems”, BEST PAPER AWARD*

PREDICTABILITY



Memory Behavior Depends on?:

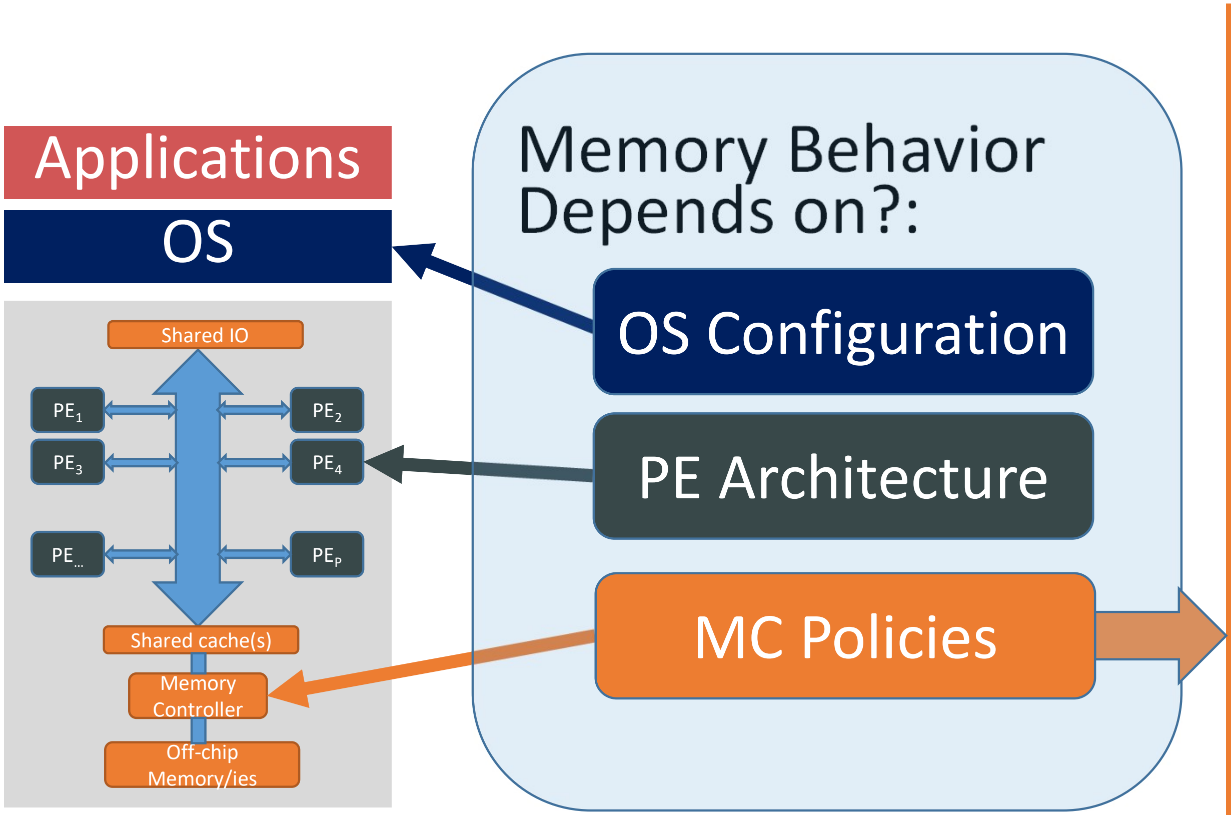
- OS Configuration
- PE Architecture
- MC Policies

System Details

MODEL

System Details

MODEL



- 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

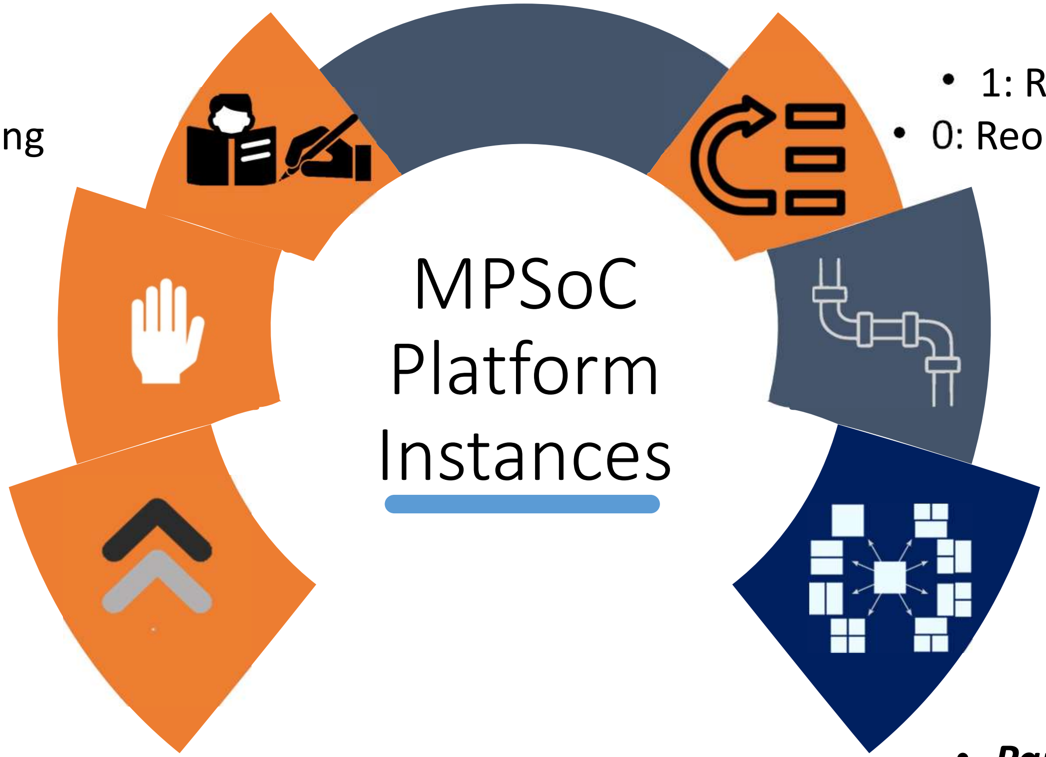
- 1: write batching
- 0: no write batching

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

R/W Reorder

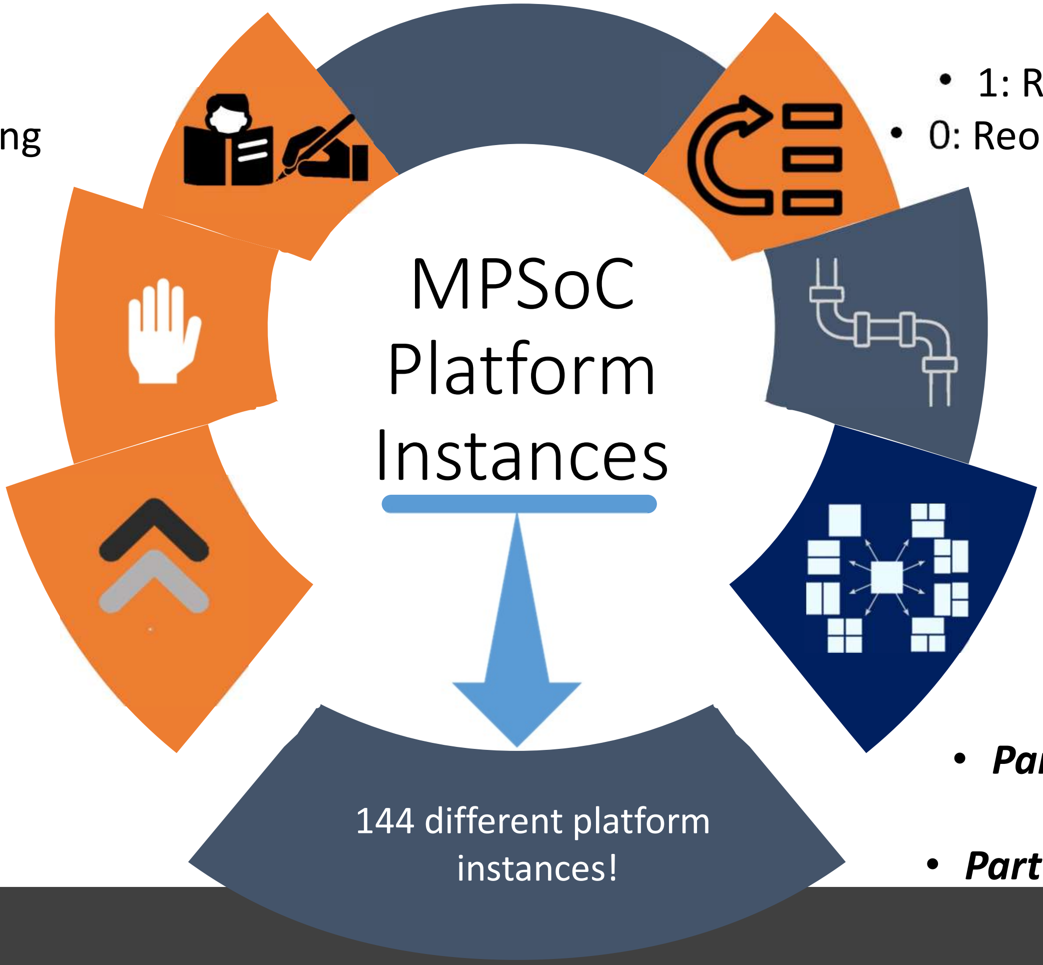
- 1: write batching
- 0: no write batching

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

144 different 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

General Observations

PREDICTABILITY

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

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

PREDICTABILITY

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												

General Observations

PREDICTABILITY

General Observations

PREDICTABILITY

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												
	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

72

<table>
<tr><th colspan="3">OS</th><th colspan="12">HW setup</th></tr>
<tr><th rowspan="2">part</th><th rowspan="2">thr</th><th rowspan="2">pr</th><th colspan="3">wb=0,breorder=0</th><th colspan="3">wb=0,breorder=1</th><th colspan="3">wb=1,breorder=0</th><th colspan="3">wb=1,breorder=1</th></tr>
<tr><th>OOO</th><th>IO-Cr</th><th>IO-All</th><th>OOO</th><th>IO-Cr</th><th>IO-All</th><th>OOO</th><th>IO-Cr</th><th>IO-All</th><th>OOO</th><th>IO-Cr</th><th>IO-All</th></tr>
<tr><td rowspan="4">Part-All</td><td>0</td><td>0</td><td></td><td></td><td></td><td rowspan="12" colspan="3">UNBOUNDED</td><td></td><td></td><td></td><td rowspan="12" colspan="3">Same as wb=1,breorder=0</td></tr>
<tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></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></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="4">No-Part</td><td>0</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></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></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="4">Part-Cr</td><td>0</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></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></td><td></td><td></td><td></td><td></td></tr>
</table>

General Observations

PREDICTABILITY

72

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									

General Observations

PREDICTABILITY

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

General Observations

PREDICTABILITY

54

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									

General Observations

PREDICTABILITY

54

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

PREDICTABILITY

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			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									

General Observations

PREDICTABILITY

General Observations

PREDICTABILITY

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

PREDICTABILITY

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	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	6
No-Part	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
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

PREDICTABILITY

34

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								
	1	0	confg9								
	1	1	confg10								

General Observations

PREDICTABILITY

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!!

34

General Observations

PREDICTABILITY

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

PREDICTABILITY

General Observations

PREDICTABILITY

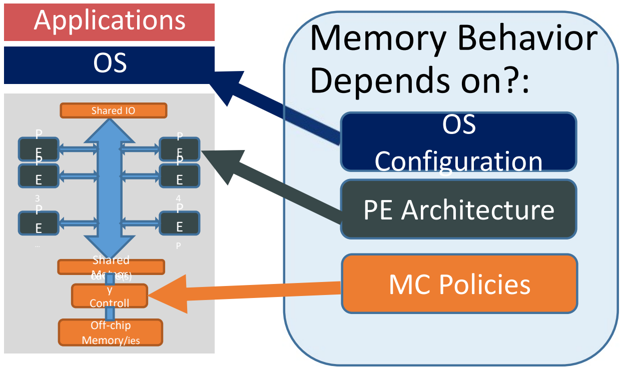
28

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

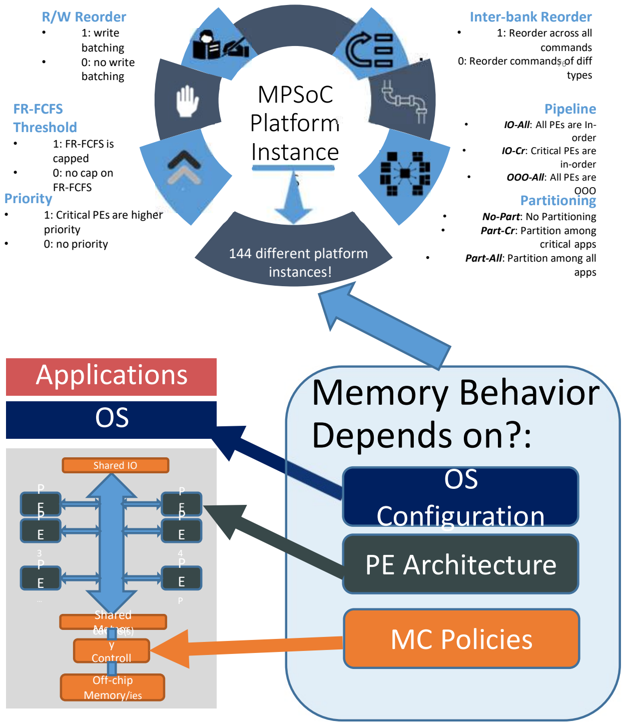
Methodology

PREDICTABILITY



Memory Behavior Depends on?:

- OS Configuration
- PE Architecture
- MC Policies



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

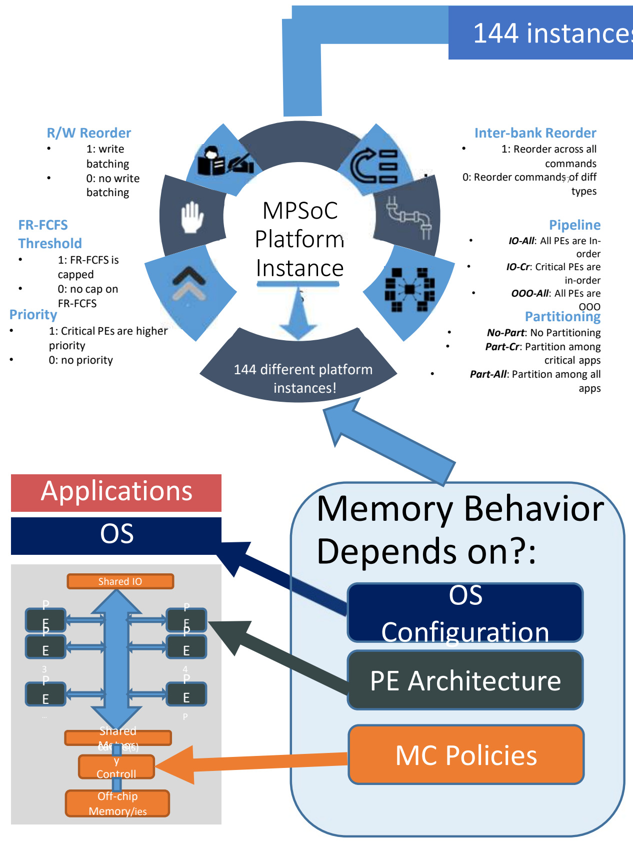
- ***No-Part***: No Partitioning
- ***Part-Cr***: Partition among critical apps
- ***Part-All***: Partition among all apps

Memory Behavior Depends on?:

- OS Configuration
- PE Architecture
- MC Policies

Methodology

PREDICTABILITY



144 instances

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 Instance

144 different 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

Applications

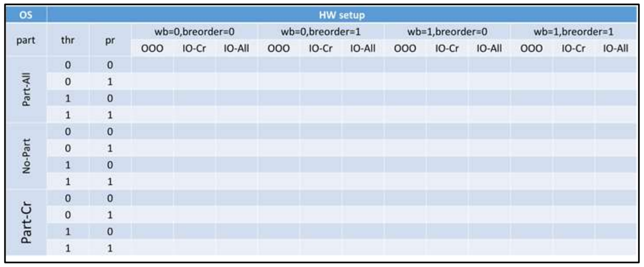
OS

Memory Behavior Depends on?:

OS Configuration

PE Architecture

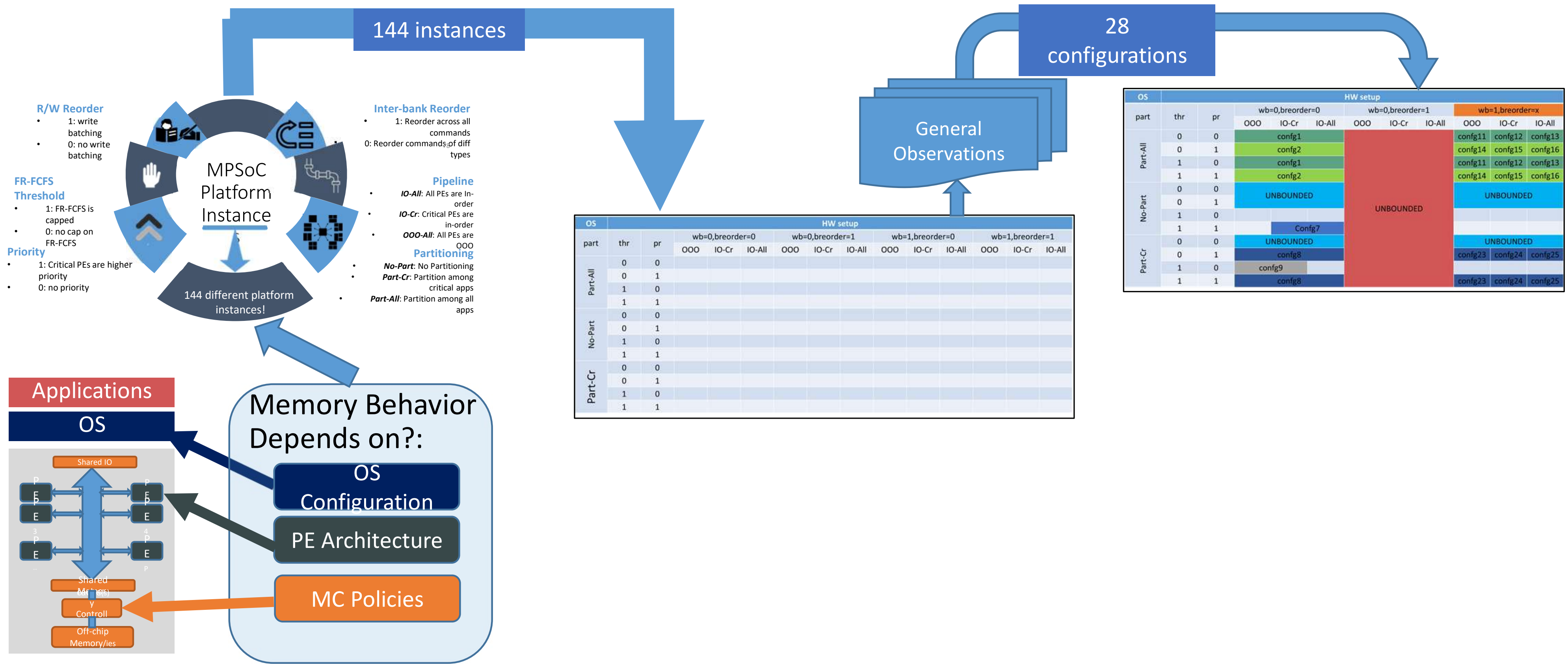
MC Policies



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												

Methodology

PREDICTABILITY



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?:

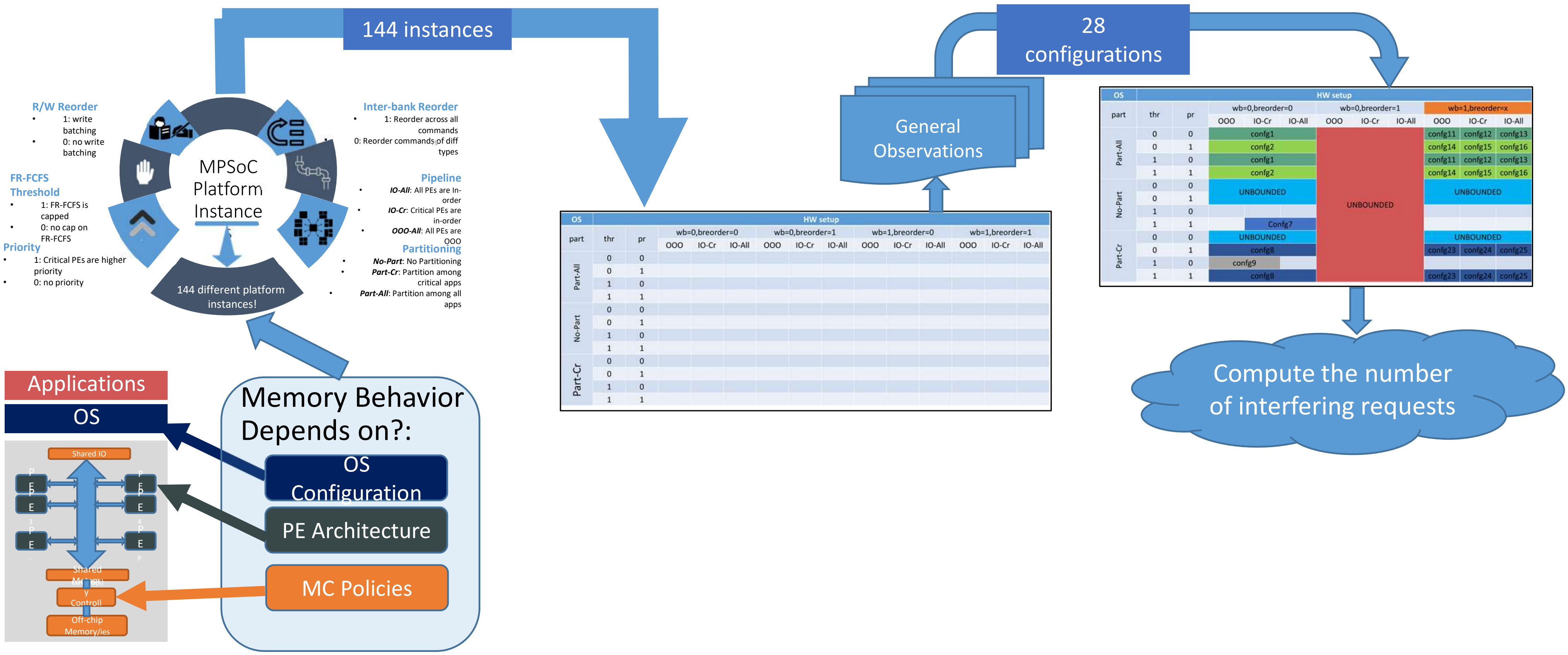
- OS Configuration
- PE Architecture
- MC Policies

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

Methodology

PREDICTABILITY



144 instances

28 configurations

General Observations

Compute the number of interfering requests

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 Instance

144 different 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

Memory Behavior Depends on?:

OS Configuration

PE Architecture

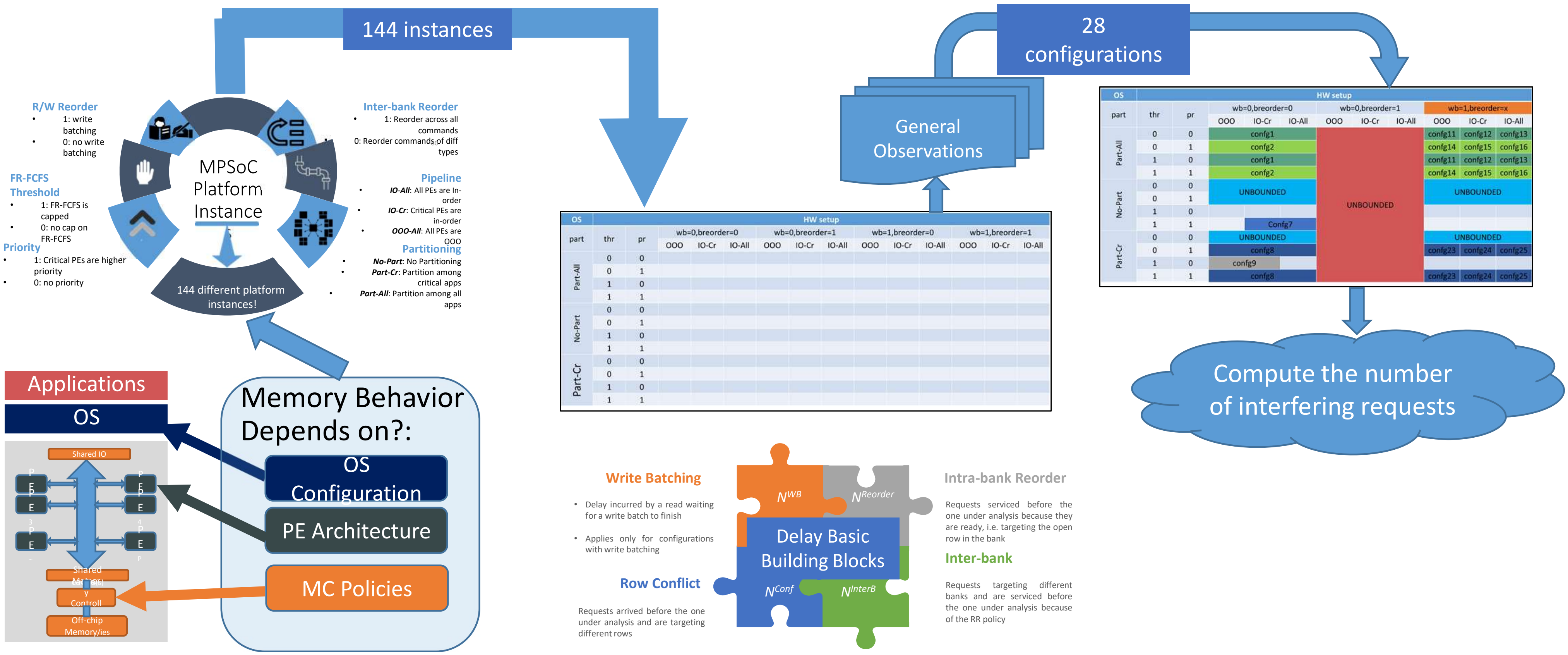
MC Policies

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

Methodology

PREDICTABILITY



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

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

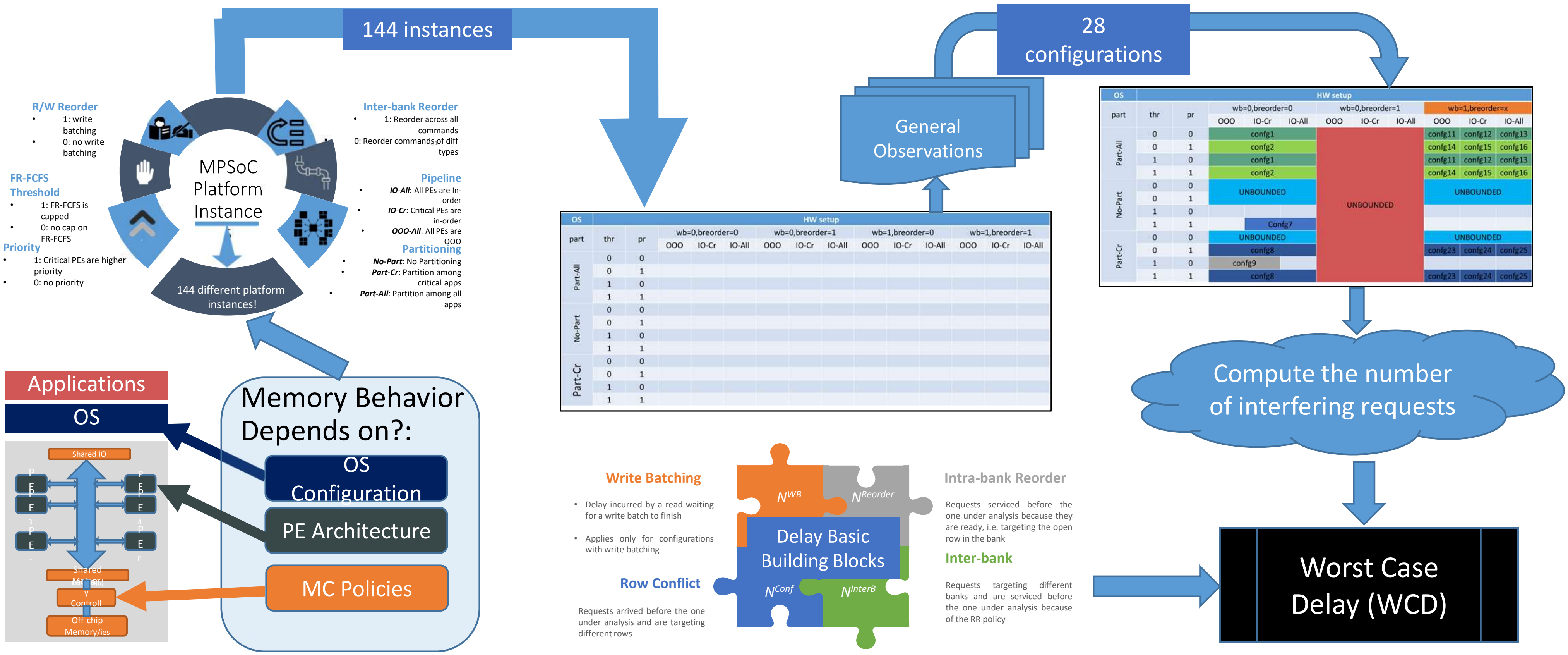
Requests serviced before the one under analysis because they are ready, i.e. targeting the open row in the bank

Inter-bank

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

Methodology

PREDICTABILITY



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 Instance

144 different 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

Applications

OS

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

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

N^{WB} $N^{Reorder}$

Delay Basic Building Blocks

N^{Conf} N^{InterB}

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

Worst Case Delay (WCD)

Methodology

PREDICTABILITY

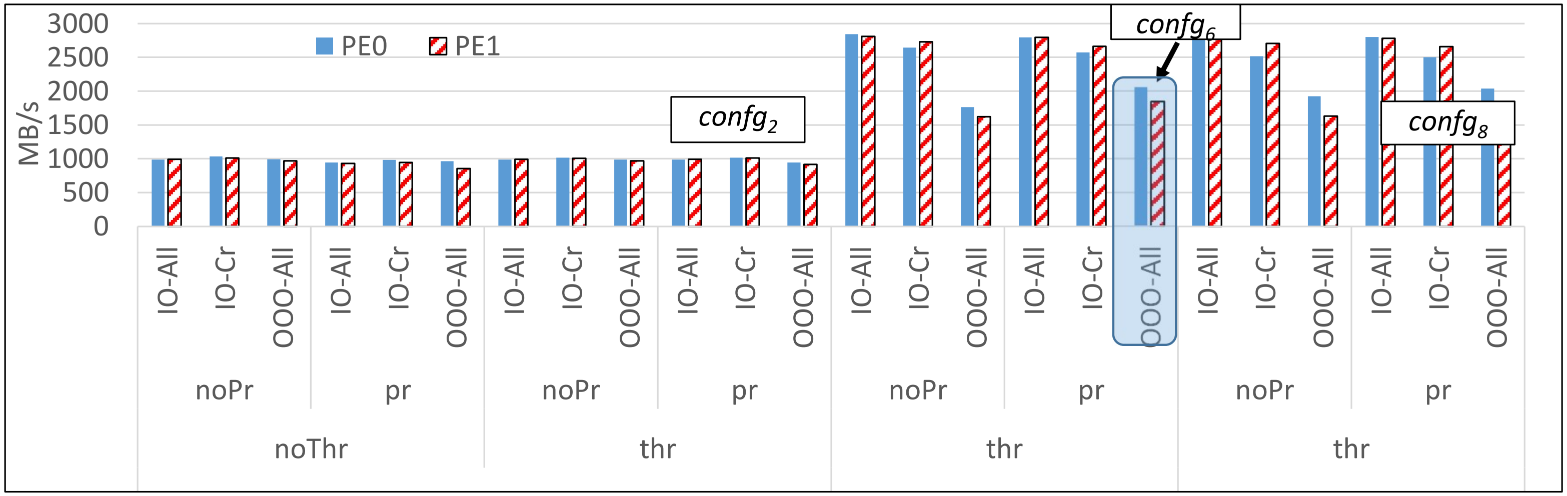
Evaluation

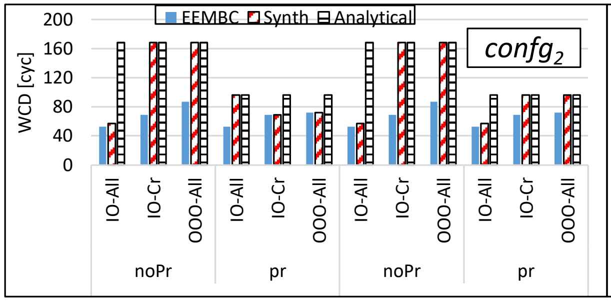
PREDICTABILITY

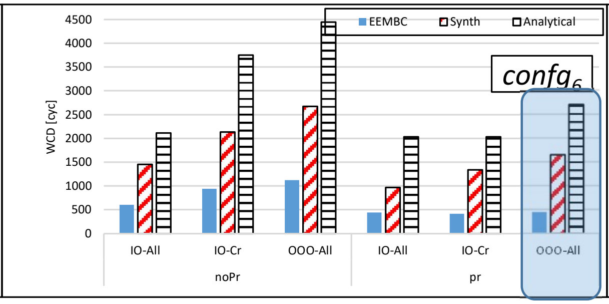
PEs	• A private 16KB L1 and a shared 1MB L2 cache • An in-order PE has a maximum of one pending request to the DRAM • An OOO PE has a maximum of 4 pending requests to the DRAM (PR = 4) • Four-processor system unless otherwise specified	
OS Mapping	• Through the virtual-to-physical address mapping component 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

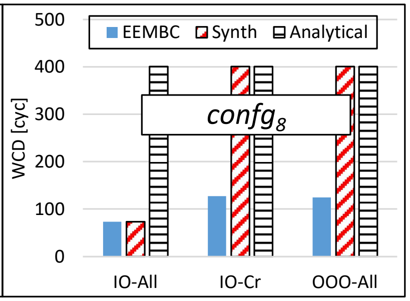
Compared to Confg 6 (No-Part):

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







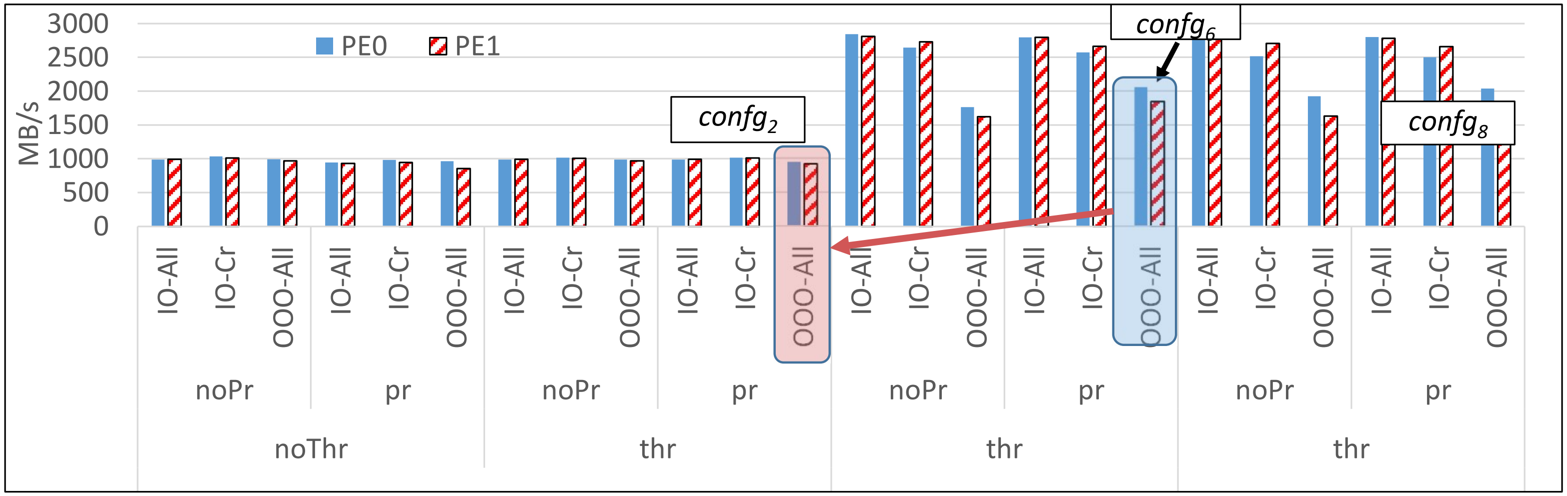


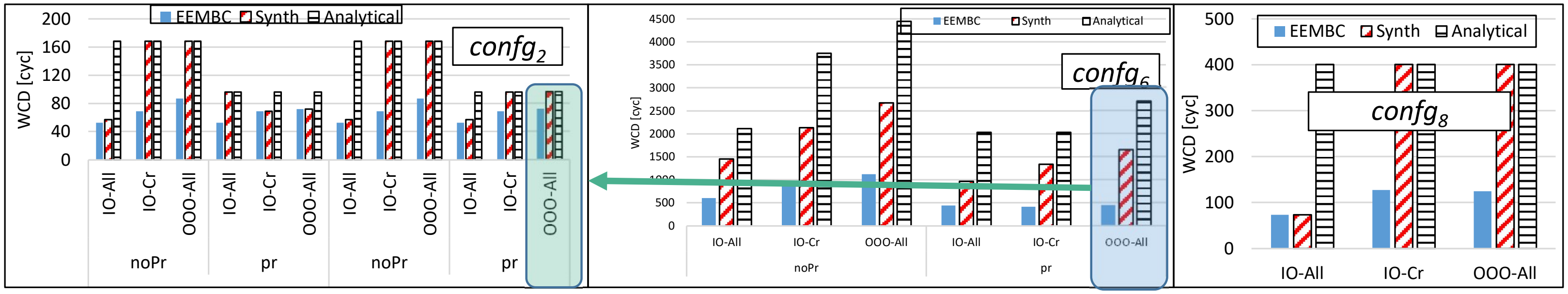
Evaluation

PREDICTABILITY

Compared to Confg 6 (No-Part):

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





Evaluation

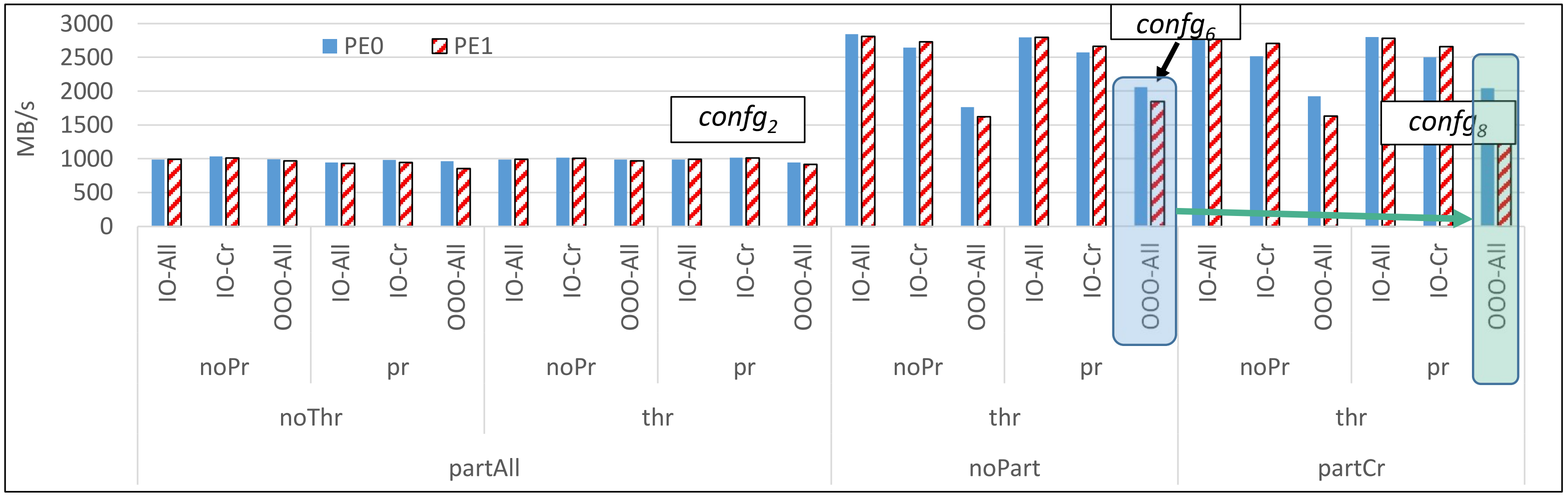
PREDICTABILITY

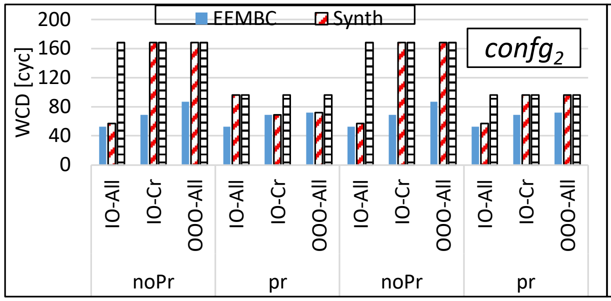
Evaluation

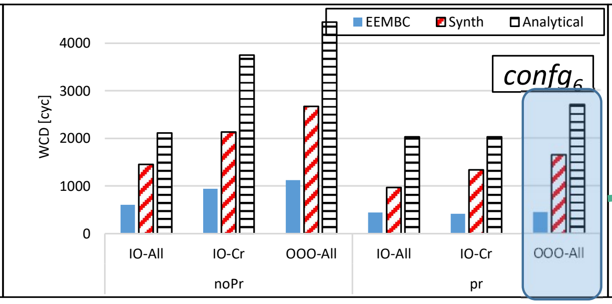
PREDICTABILITY

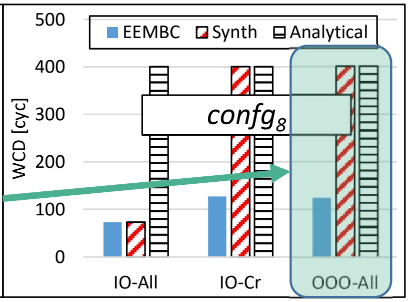
Compared to Confg 6 (No-Part):

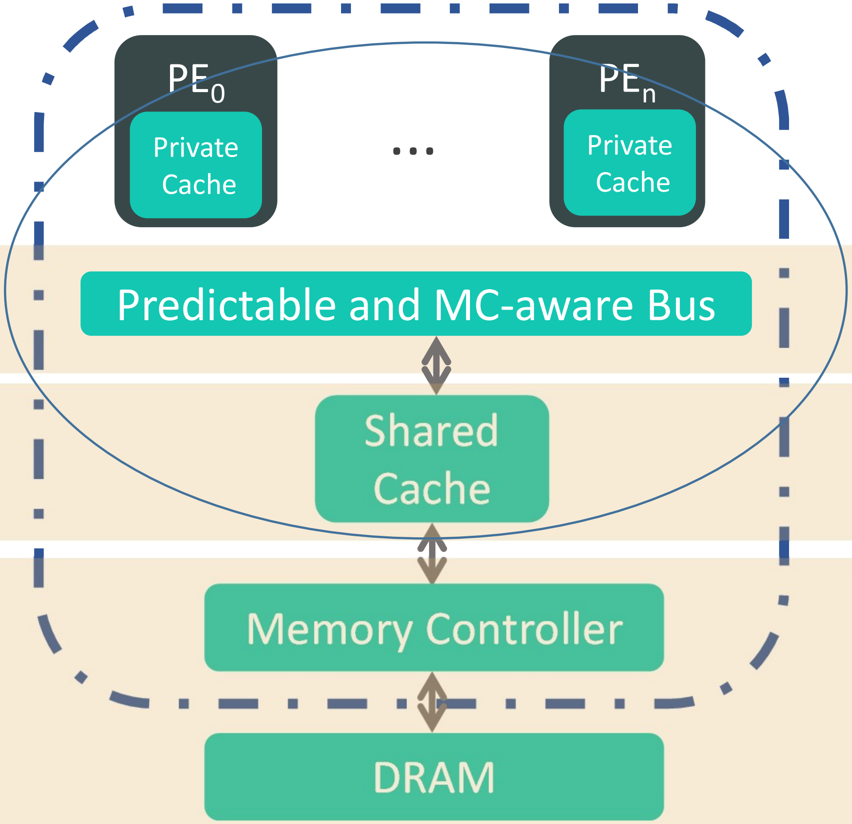
- Confg 2 (Part-All):
  - 96% less WCD
  - 60% BW degradation
- Confg 8 (Part-Cr + FP):
  - 89% less WCD
  - 0.85% BW degradation











CArb [RTAS'16]

PMSI [RTAS'17], PENDULUM [RTSS'19]
HourGlass [ArXiv'18]

PMC [RTAS'15, TECS'16]
MCSim [TECS'17]
MCS-MPSoCs [EMSOFT'18, TCAD'18]
RLDRAM [RTSS'18]
DRAMbulism [RTAS'20]

PREMIUM

Predictable Memory Hierarchy

Predictable CPSoC

PREDICTABILITY

Ignore

- Adopts an independent-task model → No communication amongst tasks

Prevent

- Enforcing complete isolation between tasks.
  - At the shared cache: strict cache partitioning and coloring
  - At the DRAM: bank privatization

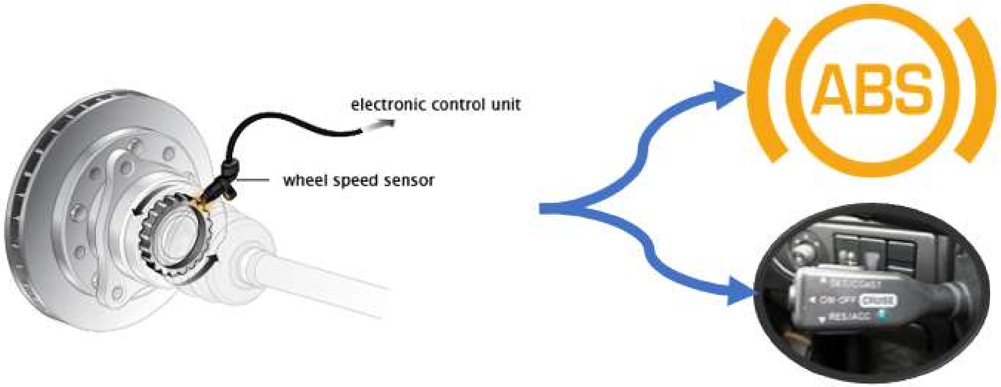
Common Approach

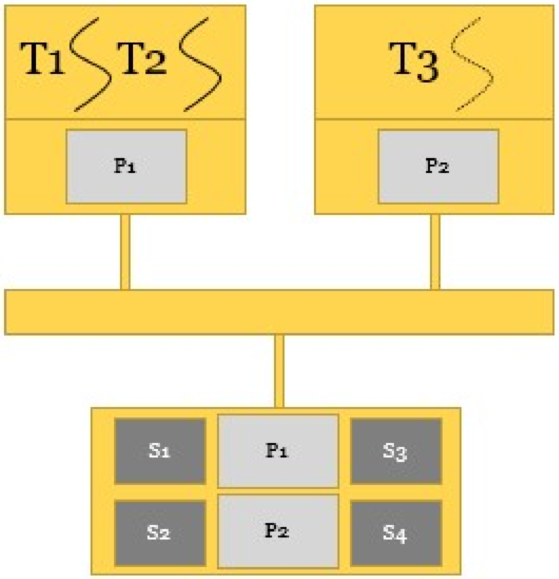
Data Sharing

Common Approach

Data Sharing

- May result in a poor memory or cache utilization
  - e.g.: a task has conflict misses, while other partitions may remain underutilized
- Does not scale with increasing number of cores
  - e.g.: number of PEs ≤ number of DRAM banks
- Not viable in emerging systems due to increased functionality and massive data



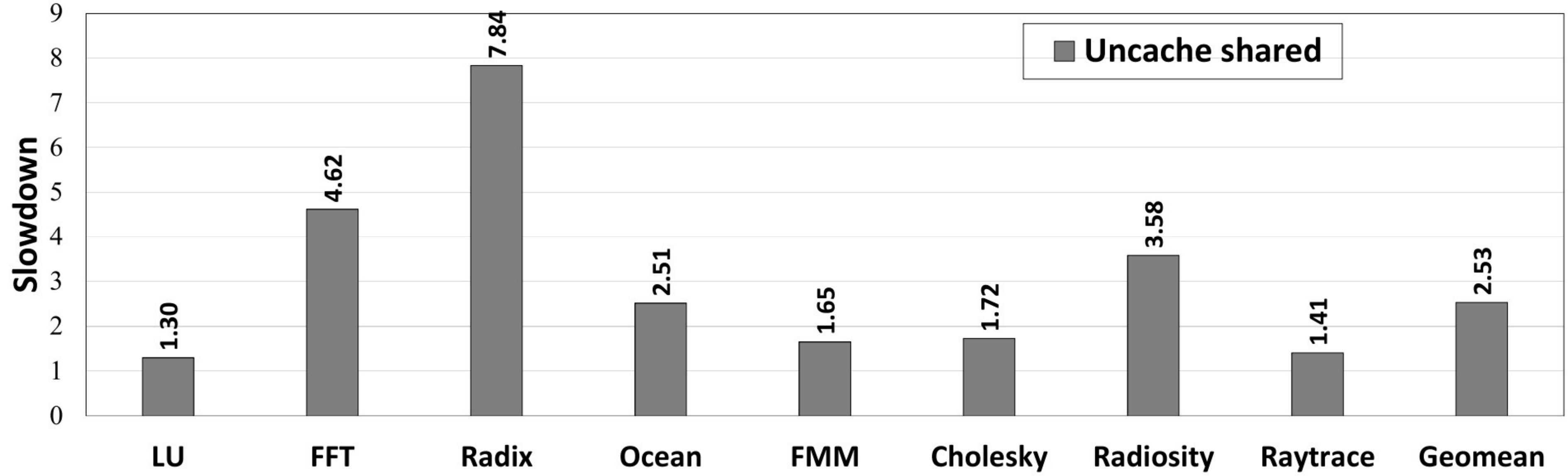


✓ Simpler timing analysis
✕ Hardware changes
✕ Long execution time

Solution:

No caching of shared data

[Hardy et al., RTSS’09]
[Lesage et al., RTNS’10]
[Bansal et al., arXiv’19]
[Chisholm et al., RTSS’16]



Solution:

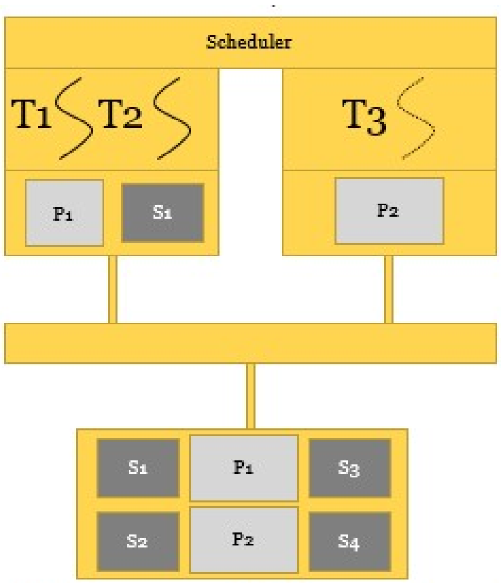
No caching of shared data

[Hardy et al., RTSS’09]

[Lesage et al., RTNS’10]

[Bansal et al., arXiv’19]

[Chisholm et al., RTSS’16]



- ✓ Private cache hits on shared data
- ✓ No hardware changes
- ✗ Limited multi-core parallelism
- ✗ Changes to OS scheduler

Another Solution:

Task scheduling on shared data

1. Scheduling tasks with shared data such that they never run in parallel [Becker et al. , ECRTS’16]

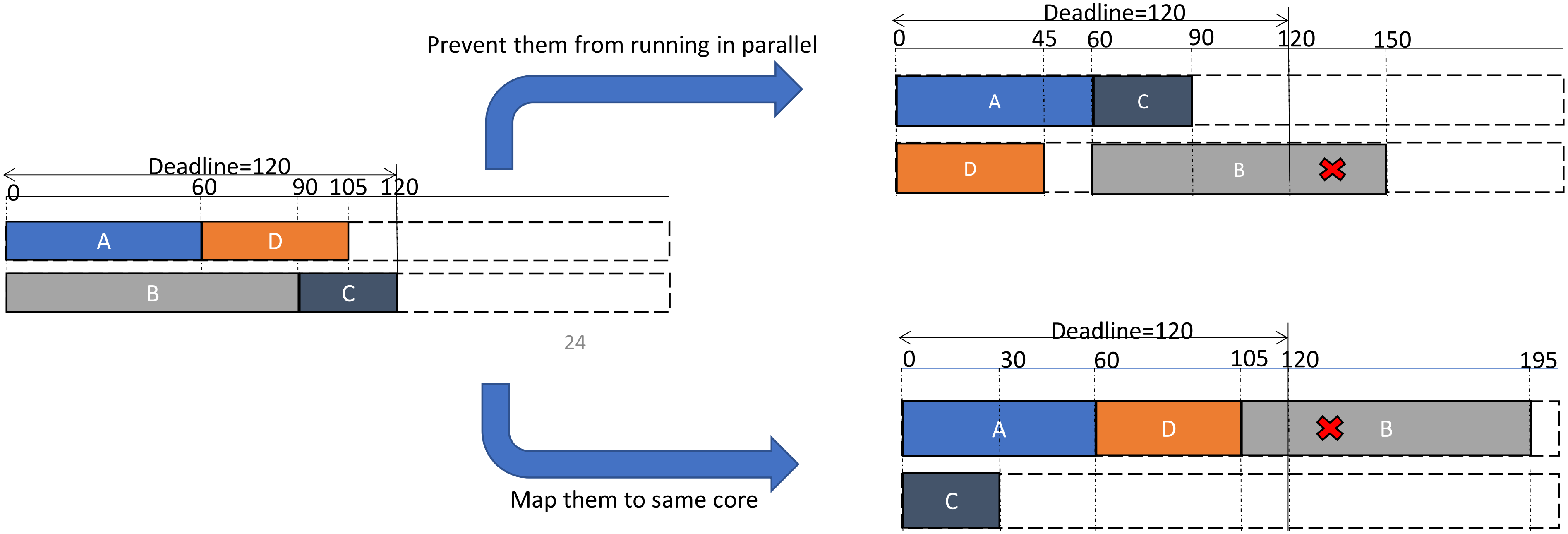
2. Assigning tasks with shared data to the same core [Chisholm et al, RTSS’16]

3. Incorporating run-time performance metrics collected through hardware counters to make data-wise scheduling decisions [Gracioli et al., SIGOPS’15]

Another Solution:

Task scheduling on shared data

Example: B shares data with A and C



1. Scheduling tasks with shared data such that they never run in parallel [Becker et al. , ECRTS'16]
2. Assigning tasks with shared data to the same core [Chisholm et al, RTSS'16]
3. Incorporating run-time performance metrics collected through hardware counters to make data-wise scheduling decisions [Gracioli et al., SIGOPS'15]

The mainstream solution is to provide shared memory and prevent incoherence through a hardware cache coherence protocol, making caches functionally invisible to software.

DOI:10.1145/2209249.2209269

On-chip hardware coherence can scale gracefully as the number of cores increases.

BY MILO M.K. MARTIN, MARK D. HILL, AND DANIEL J. SORIN

Why On-Chip Cache Coherence Is Here to Stay

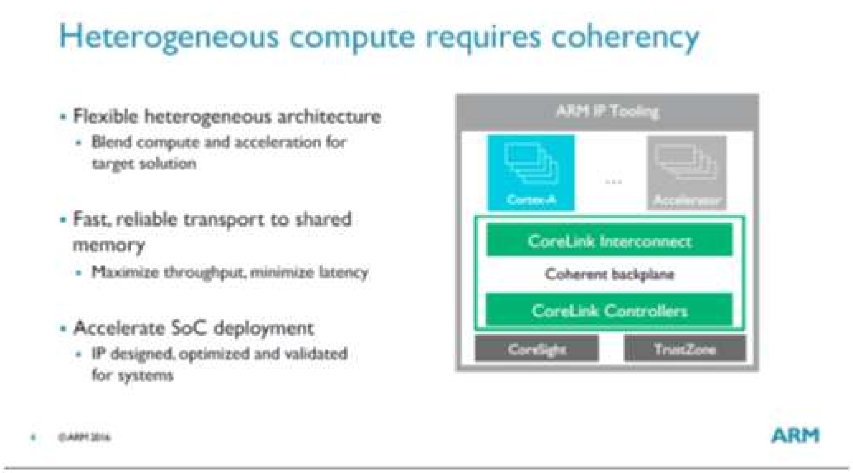
SHARED MEMORY IS the dominant low-level communication paradigm in today's mainstream multicore processors. In a shared-memory system, the (processor) cores communicate via loads and stores to a shared address space. The cores use caches to reduce the average memory latency and memory traffic. Caches are thus beneficial, but private caches lead to the possibility of cache incoherence. The mainstream solution is to provide shared memory and prevent incoherence through a hardware cache coherence protocol, making caches functionally invisible to software. The incoherence problem and basic hardware coherence solution are outlined in the sidebar, "The Problem of Incoherence," page 86.

Cache-coherent shared memory is provided by mainstream servers, desktops, laptops, and mobile devices and is available from all major vendors, including AMD, ARM, IBM, Intel, and Oracle (Sun).

78 COMMUNICATIONS OF THE ACM | JULY 2012 | VOL. 55 | NO. 7

Coherence is the norm in COTS platforms

Data Sharing



Heterogeneous compute requires coherency

- Flexible heterogeneous architecture
  - Blend compute and acceleration for target solution
- Fast, reliable transport to shared memory
  - Maximize throughput, minimize latency
- Accelerate SoC deployment
  - IP designed, optimized and validated for systems

ARM

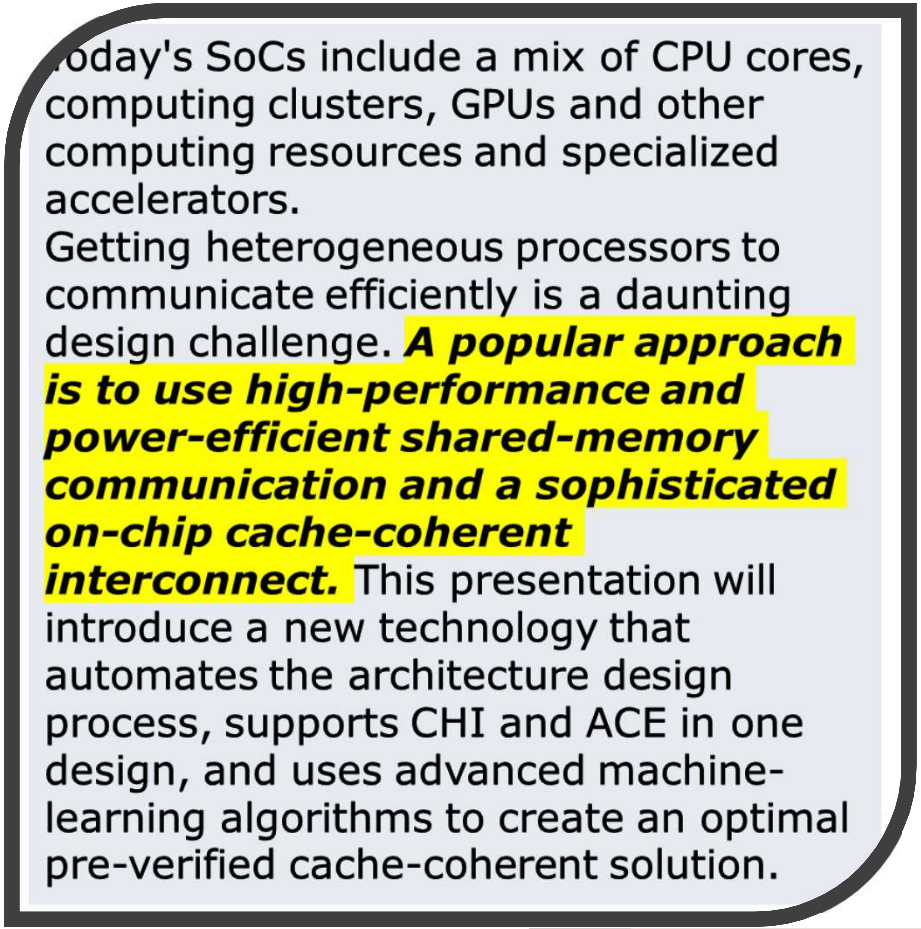
Coherence is the Industry’s Choice

Data Sharing

Coherency: The New Normal in SoCs

Anush Mohandass

anush@netspeedsystems.com



Today's SoCs include a mix of CPU cores, computing clusters, GPUs and other computing resources and specialized accelerators.
Getting heterogeneous processors to communicate efficiently is a daunting design challenge. ***A popular approach is to use high-performance and power-efficient shared-memory communication and a sophisticated on-chip cache-coherent interconnect.*** This presentation will introduce a new technology that automates the architecture design process, supports CHI and ACE in one design, and uses advanced machine-learning algorithms to create an optimal pre-verified cache-coherent solution.

Coherence is the Industry’s Choice

Data Sharing

ARTERIS IP

Enabling Mixed-Protocol Heterogeneous Cache Coherency and ISO 26262 Functional Safety

LINLEY PROCESSOR CONFERENCE
4 OCTOBER 2017

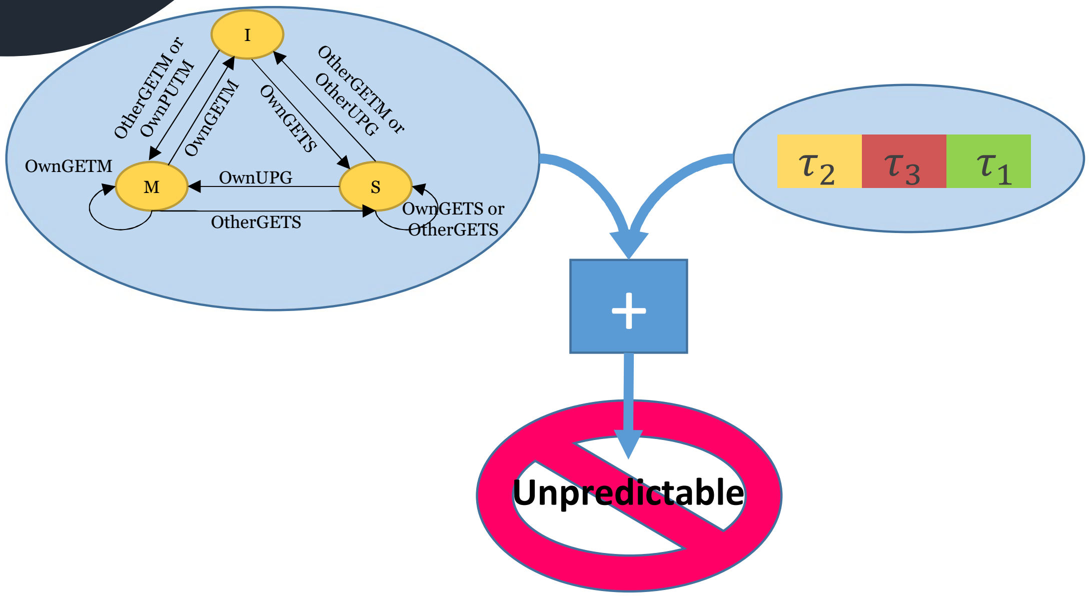
The Linley Group

SANJAY DESHPANDE
Director of Cache Coherency and
Interconnect Architecture

Autonomous driving requirements are mandating the simultaneous use of multiple types of processing units to efficiently execute sophisticated image processing, sensor fusion, and machine learning/AI algorithms.
This presentation introduces ***new coherency platform technology that enables the integration of heterogeneous cache coherent hardware accelerators and CPUs***, using a mixture of ARM ACE, CHI, and CHI Issue B protocols, into systems that meet both the requirements of high compute performance and ISO 26262-compliant functional safety.

Coherence is the Industry’s Choice

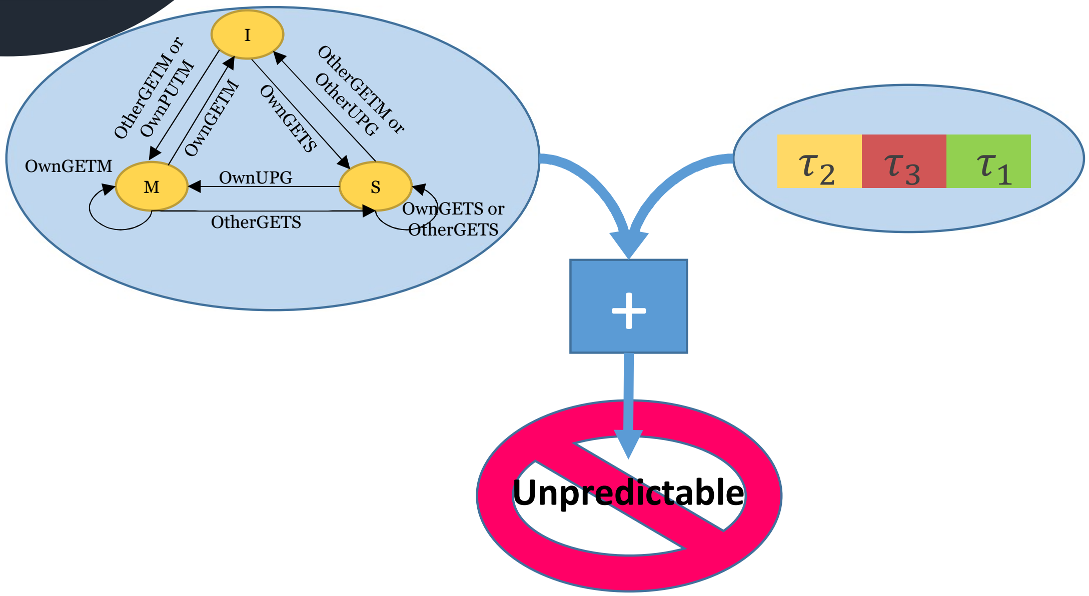
Data Sharing



Unpredictability in Sharing Data

*[RTAS'17] **Mohamed Hassan**, Anirudh M. Kaushik, Hiren Patel, "Predictable Cache Coherence for Multi-Core Real-Time Systems"*

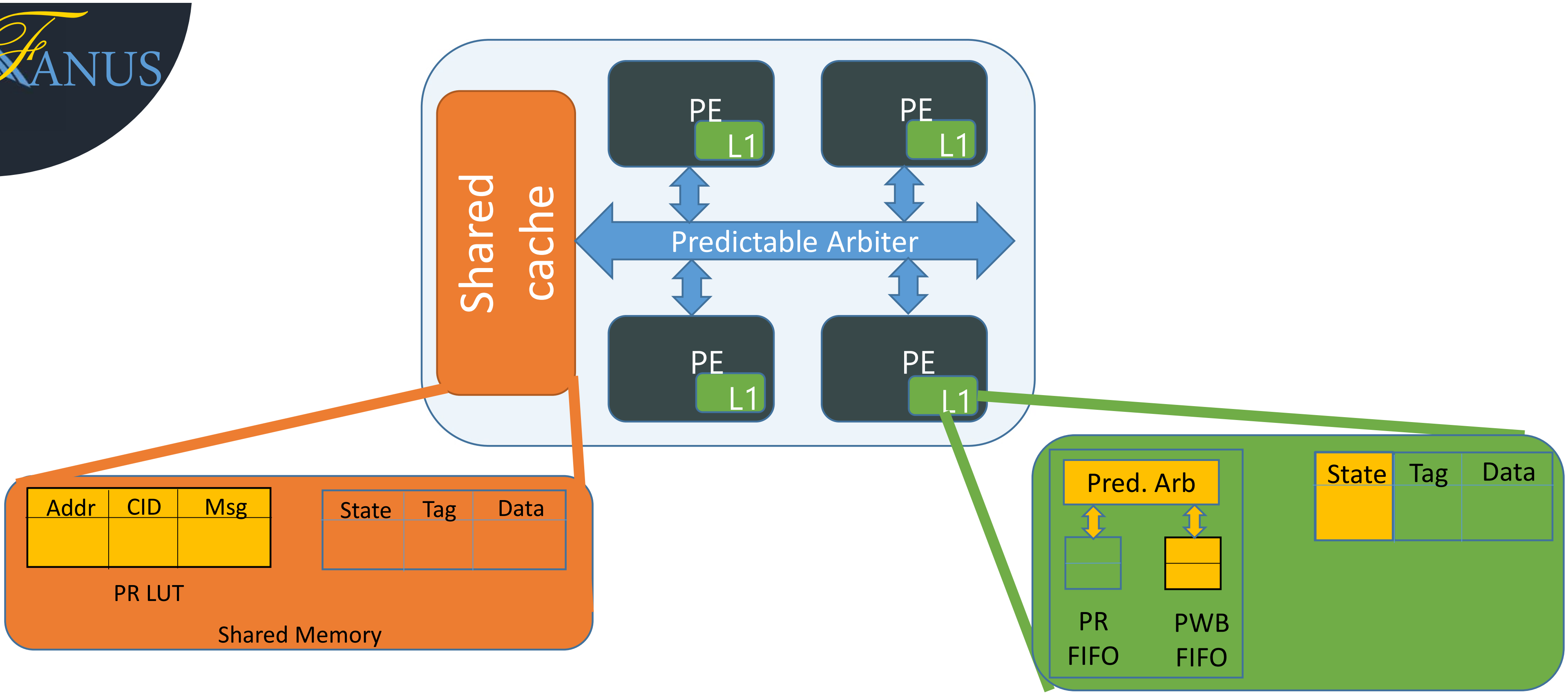
Data Sharing



- Inter-core coherence interference on same cache line
- Inter-core coherence interference on different cache lines
- Inter-core coherence interference due to write hits
- Intra-core coherence interference

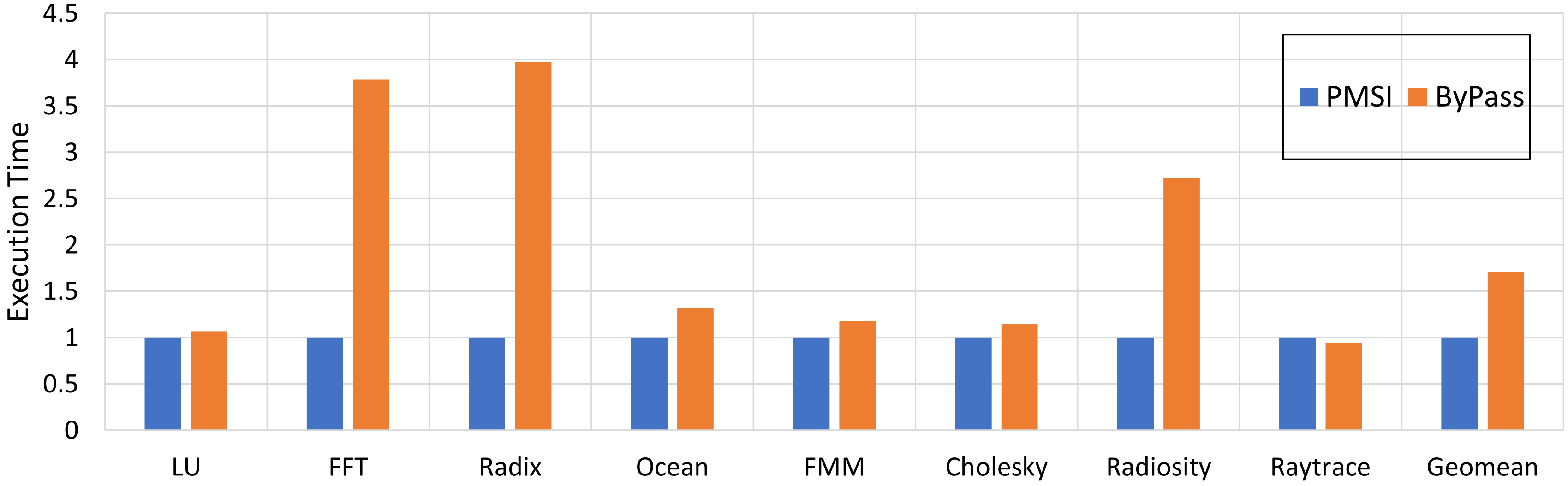
Unpredictability in Sharing Data

Data Sharing



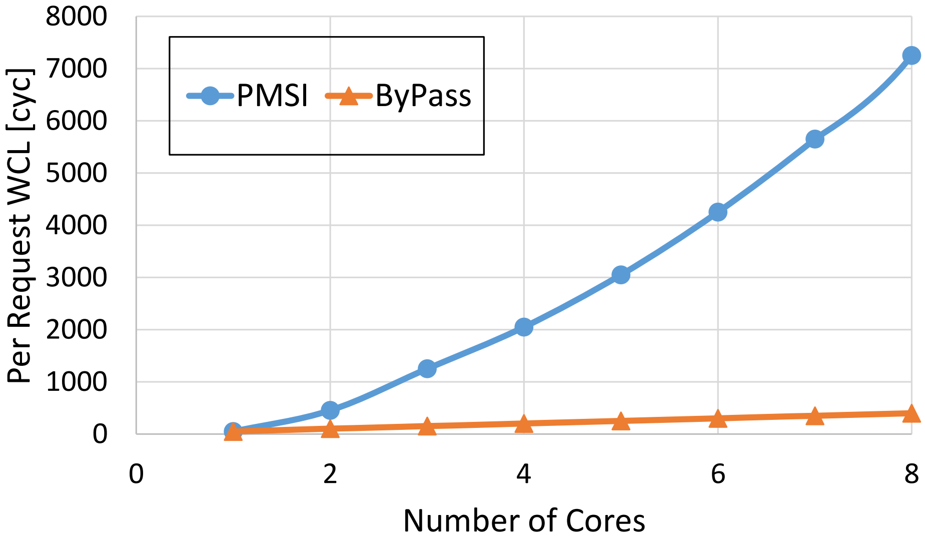
PMSI: Predictable Cache Coherence

Data Sharing



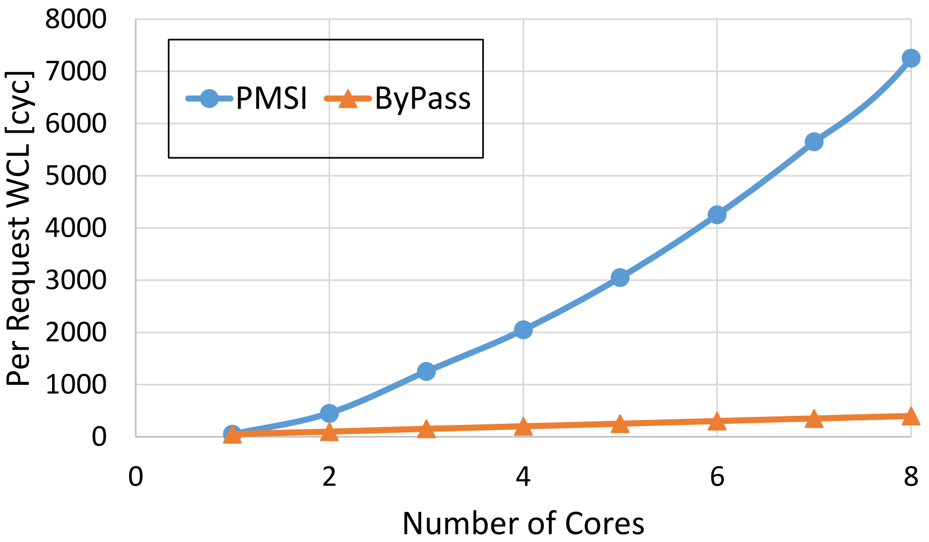
Benefit of Coherence: Up to 3x performance

Data Sharing



Problem of PMSI: Coherence effect on WC

Data Sharing

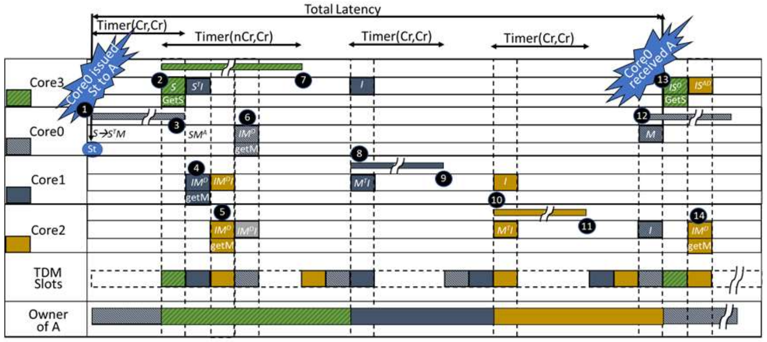


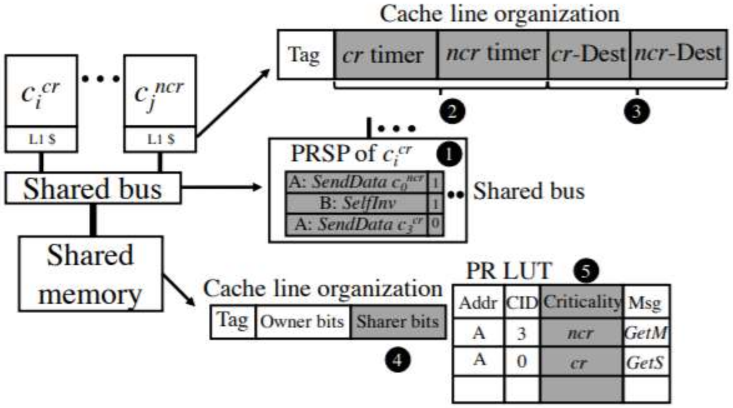
How do we improve over that?
Do all cores have to suffer this high WCL? → Differentiated-Service

Problem of PMSI: Coherence effect on WC

Data Sharing

- **Time-based Cache Coherence**
    - Configurable timers for critical/non-critical cores
- **Fixed Priority Arbitration**
    - If both critical and non-critical requesting same cache line → critical gets it
- **Allows for simultaneous data sharing**
    - Both intra- and inter-criticality
- **improves WCL for critical cores while improving the BW of non-critical cores**



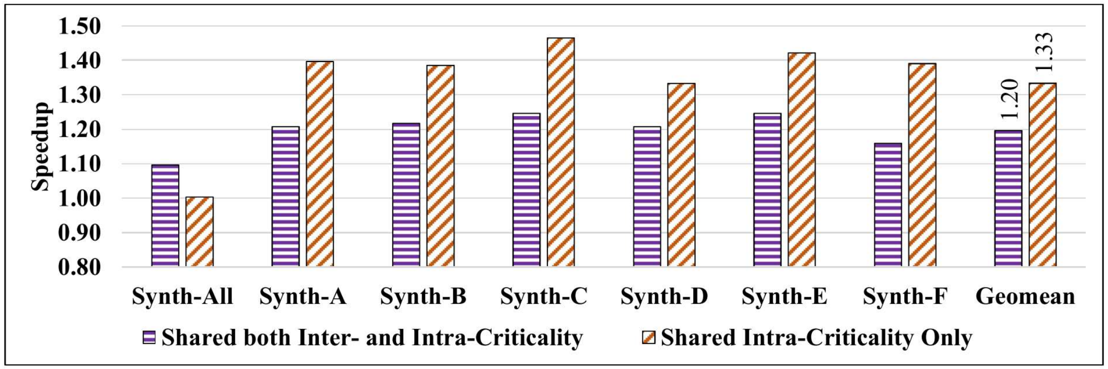


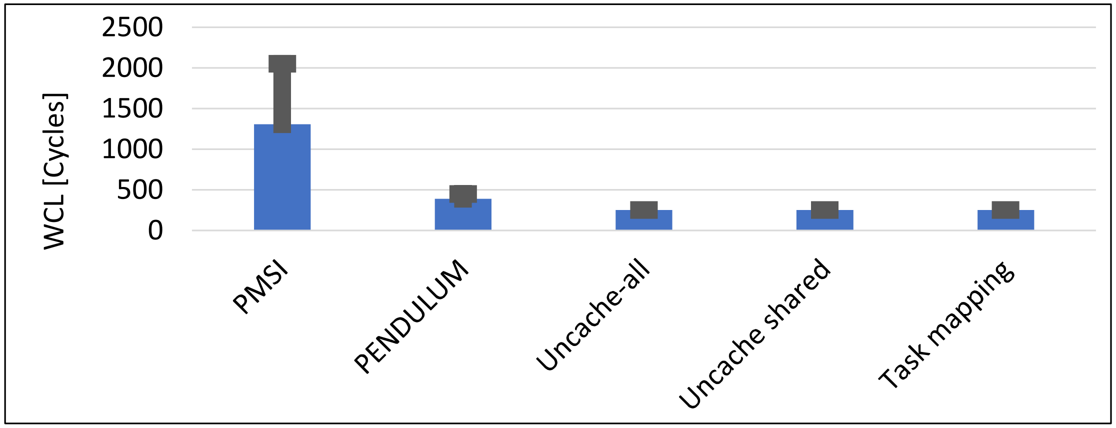
Addr	CID	Criticality	Msg
A	3	*ncr*	*GetM*
A	0	*cr*	*GetS*

PENDULUM: Cache Coherence for MCS

*[RTSS'19] Nivedita Sritharan, Anirudh M. Kaushik, **Mohamed Hassan**, Hiren Patel, "Predictable Cache Coherence for Multi-Core Real-Time Systems"*

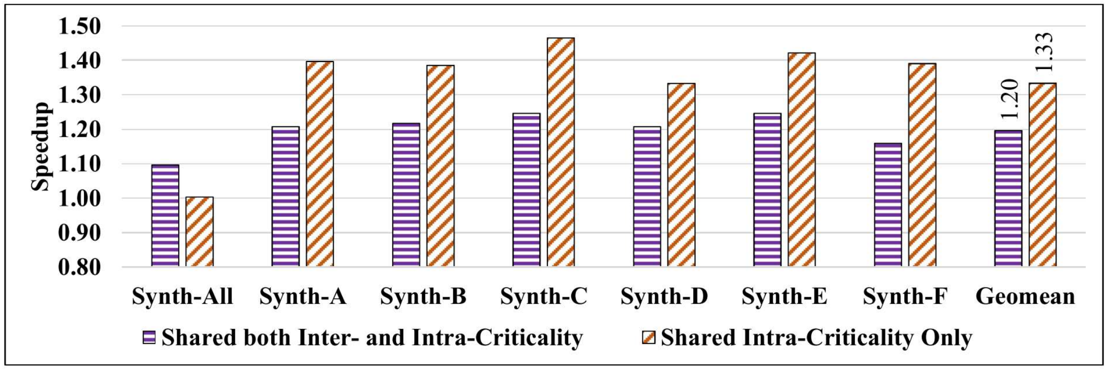
Data Sharing



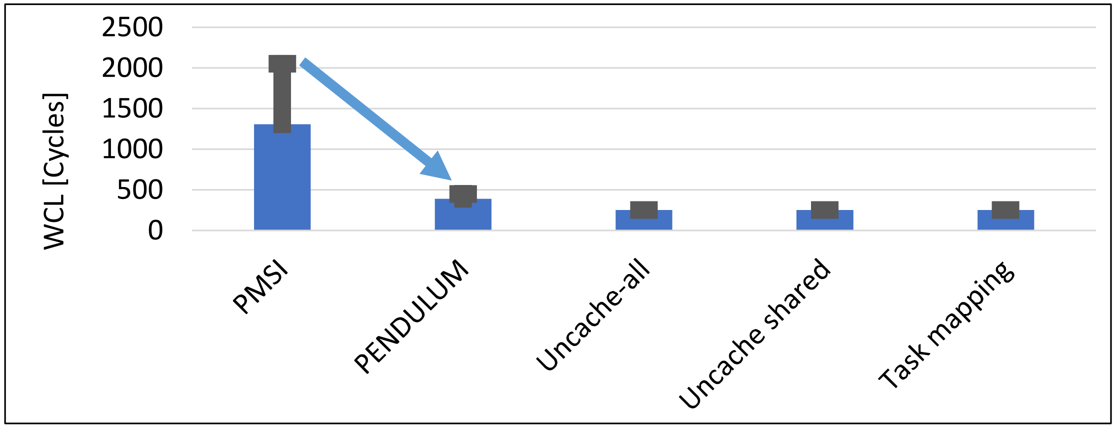


PENDULUM: Cache Coherence for MCS

Data Sharing



Close the WCL gap for critical cores



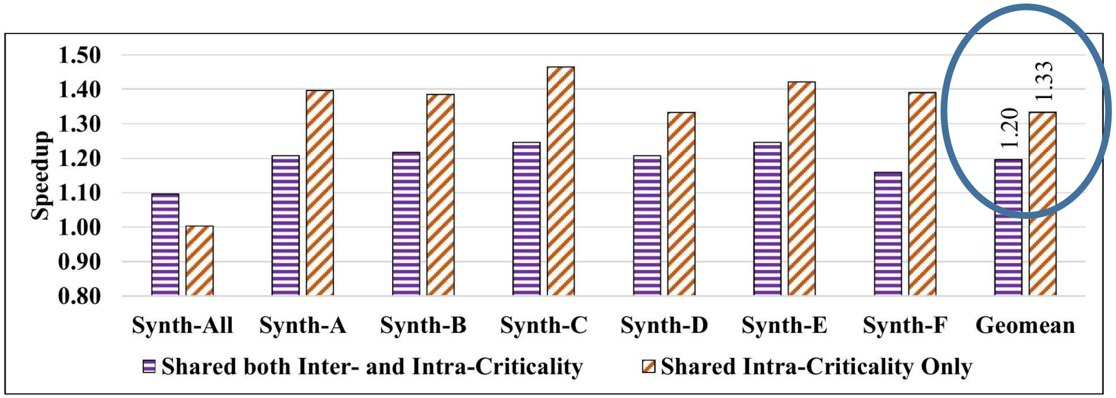
PENDULUM: Cache Coherence for MCS

Data Sharing

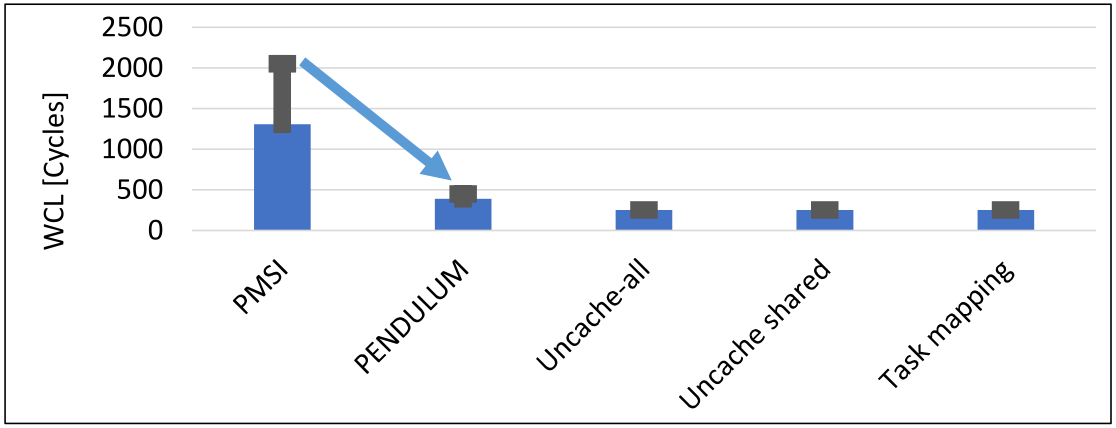
PENDULUM: Cache Coherence for MCS

Data Sharing

Maintains overall performance benefits of coherence



Close the WCL gap for critical cores



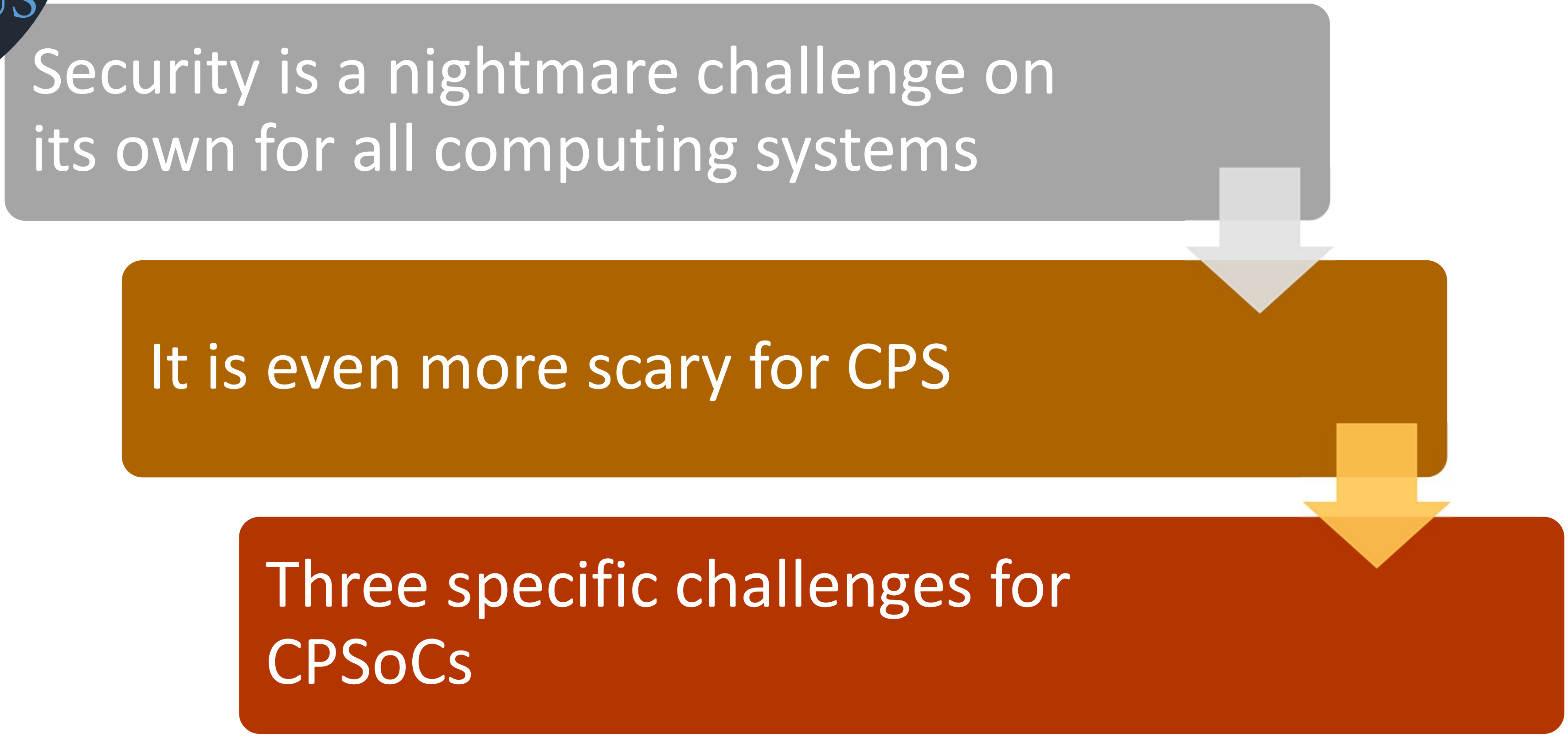
Towards Predictable, ***Secure***, and Verified CPSoCs

Security

Security is a nightmare challenge on its own for all computing systems

Three specific challenges for CPSoCs

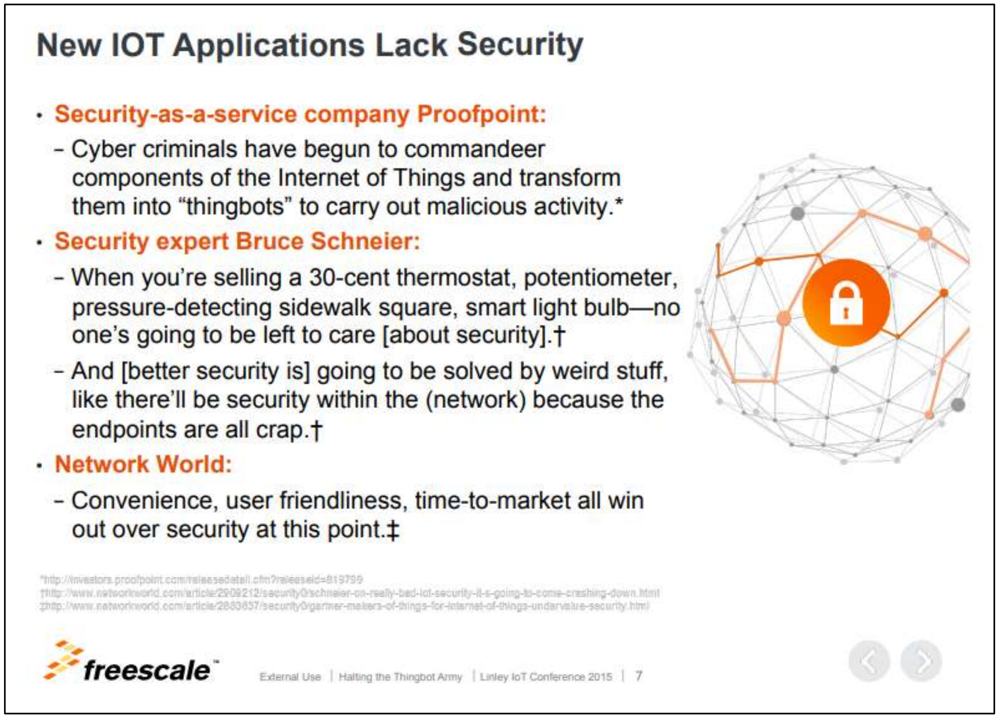
Security



Security is a nightmare challenge on its own for all computing systems

It is even more scary for CPS

Three specific challenges for CPSoCs

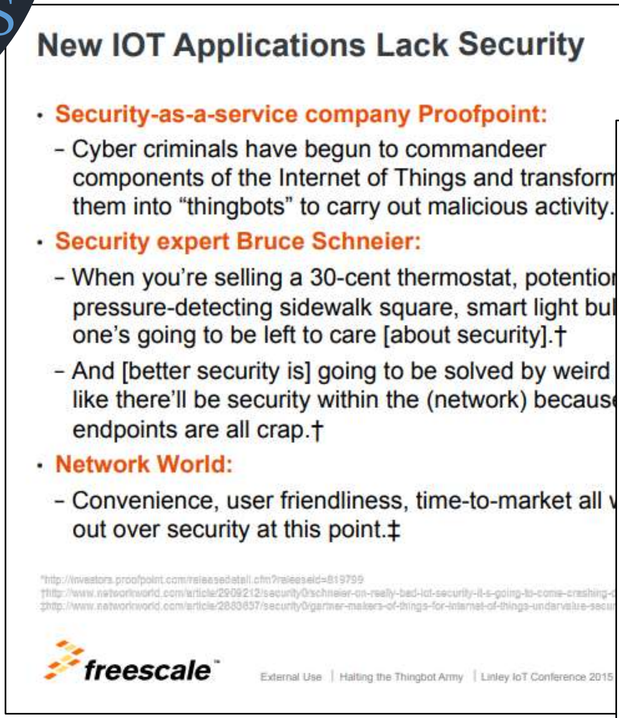


New IOT Applications Lack Security

- **Security-as-a-service company Proofpoint:**
  - Cyber criminals have begun to commandeer components of the Internet of Things and transform them into “thingbots” to carry out malicious activity.*
- **Security expert Bruce Schneier:**
  - When you’re selling a 30-cent thermostat, potentiometer, pressure-detecting sidewalk square, smart light bulb—no one’s going to be left to care [about security].†
  - And [better security is] going to be solved by weird stuff, like there’ll be security within the (network) because the endpoints are all crap.†
- **Network World:**
  - Convenience, user friendliness, time-to-market all win out over security at this point.‡

*http://investors.proofpoint.com/releasedetail.cfm?releaseid=819799
†http://www.networkworld.com/article/2909212/security0/schneier-on-really-bad-iot-security-it-s-going-to-come-crashing-down.html
‡http://www.networkworld.com/article/2883637/security0/gartner-makers-of-things-for-internet-of-things-undervalue-security.html

freescale™ External Use | Halting the Thingbot Army | Linley IoT Conference 2015 | 7

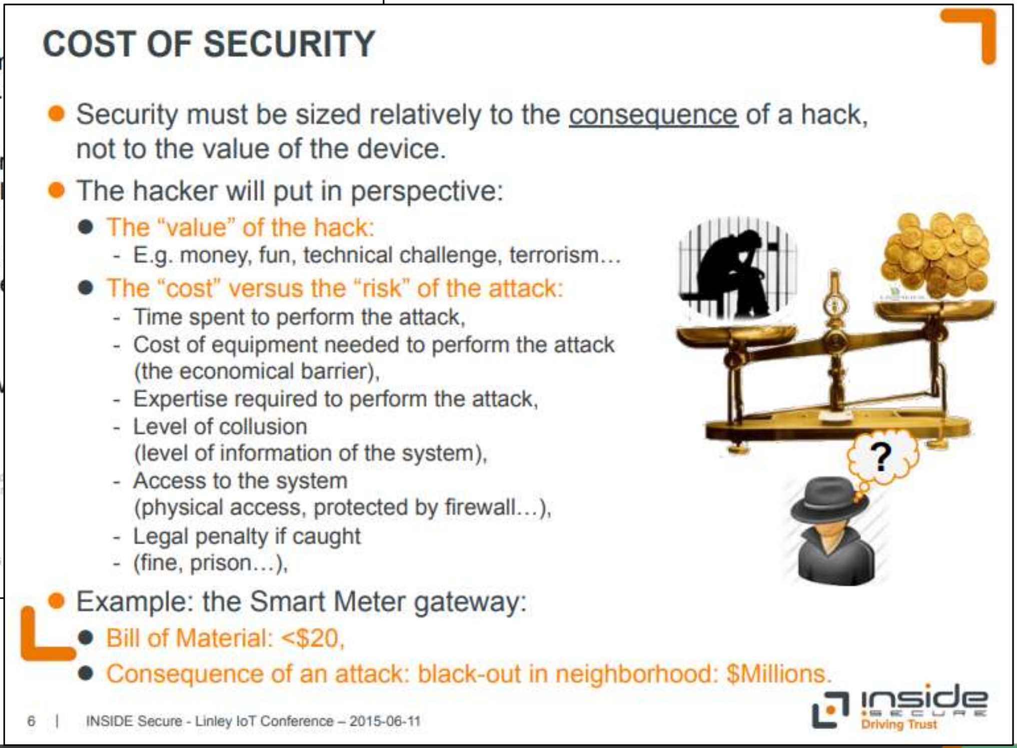


New IOT Applications Lack Security

- **Security-as-a-service company Proofpoint:**
  - Cyber criminals have begun to commandeer components of the Internet of Things and transform them into "thingbots" to carry out malicious activity.
- **Security expert Bruce Schneier:**
  - When you're selling a 30-cent thermostat, potentior pressure-detecting sidewalk square, smart light bul one's going to be left to care [about security].†
  - And [better security is] going to be solved by weird like there'll be security within the (network) becaus endpoints are all crap.†
- **Network World:**
  - Convenience, user friendliness, time-to-market all v out over security at this point.‡

*http://investors.proofpoint.com/releasedetail.cfm?releaseid=819799
†http://www.networkworld.com/article/2909212/security0/schneier-on-really-bad-iot-security-it-s-going-to-come-crashing-d
‡http://www.networkworld.com/article/2883837/security0/gartner-makers-of-things-for-internet-of-things-undervalue-secur

freescale™ External Use | Halting the Thingbot Army | Linley IoT Conference 2015



COST OF SECURITY

- Security must be sized relatively to the consequence of a hack, not to the value of the device.
- The hacker will put in perspective:
  - The "value" of the hack:
    - E.g. money, fun, technical challenge, terrorism…
  - The "cost" versus the "risk" of the attack:
    - Time spent to perform the attack,
    - Cost of equipment needed to perform the attack (the economical barrier),
    - Expertise required to perform the attack,
    - Level of collusion (level of information of the system),
    - Access to the system (physical access, protected by firewall…),
    - Legal penalty if caught
    - (fine, prison…),
- Example: the Smart Meter gateway:
  - Bill of Material: <$20,
  - Consequence of an attack: black-out in neighborhood: $Millions.

6 | INSIDE Secure - Linley IoT Conference – 2015-06-11

inside SECURE Driving Trust

Security

Denial-of-Service (DoS) Attacks [Moscibroda and Mutlu, USENIX’07]

Error Injection Attacks [Kim et al., ISCA’14]

Covert Channel Attacks [Wang et al., HPCA’14]

Side Channel Attacks [Wang et al., HPCA’14]

MCReverse

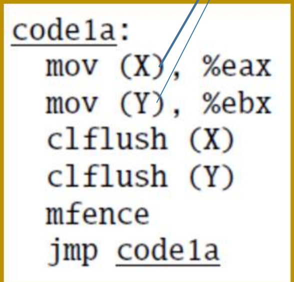
*[RTAS’15] **Mohamed Hassan**, Anirudh M. Kaushik, Hiren Patel, "Reverse Engineering Embedded Memory Controllers through Latency-based Analysis",*

*[TECS’18] **Mohamed Hassan**, Anirudh M. Kaushik, Hiren Patel, " Exposing Implementation Details of Embedded Memory Controllers through Latency-based Analysis"*

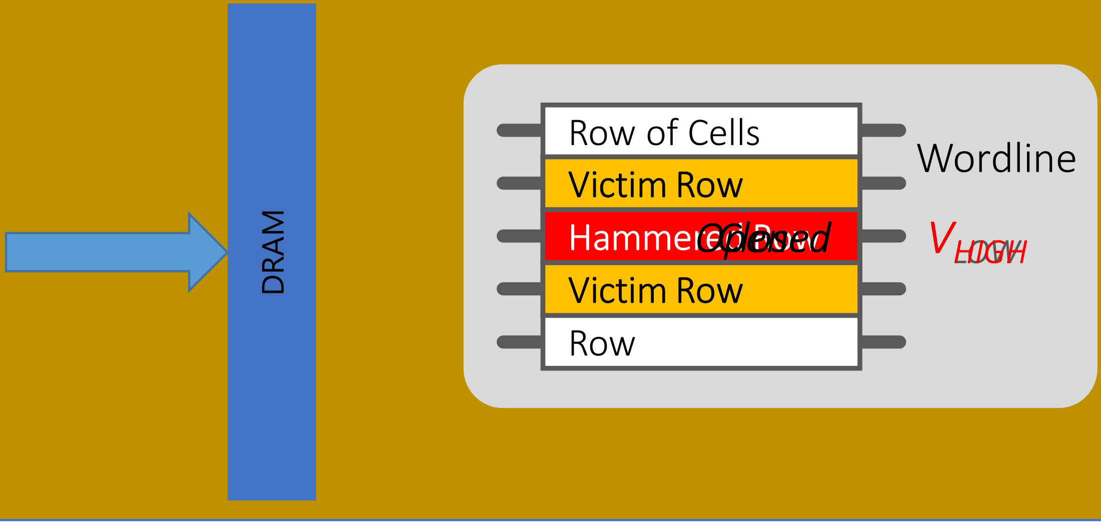
Security

Error Injection Attacks (*rowhammer*)

Targeting different pages in same bank



```
code1a:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp code1a
```

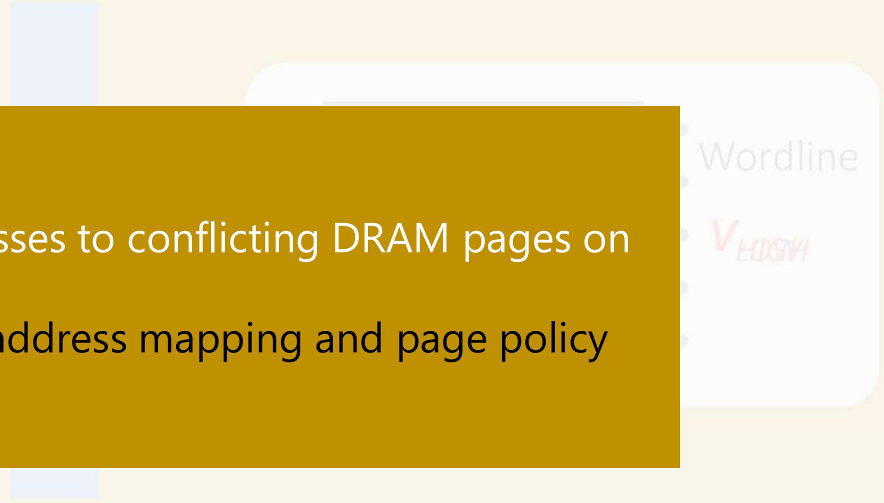


Repeatedly opening and closing a row enough times within a refresh interval induces **disturbance errors in adjacent rows** in most real DRAM chips you can buy today

Error Injection Attacks
(*rowhammer*)

Error Injection Attacks
(rowhammer)

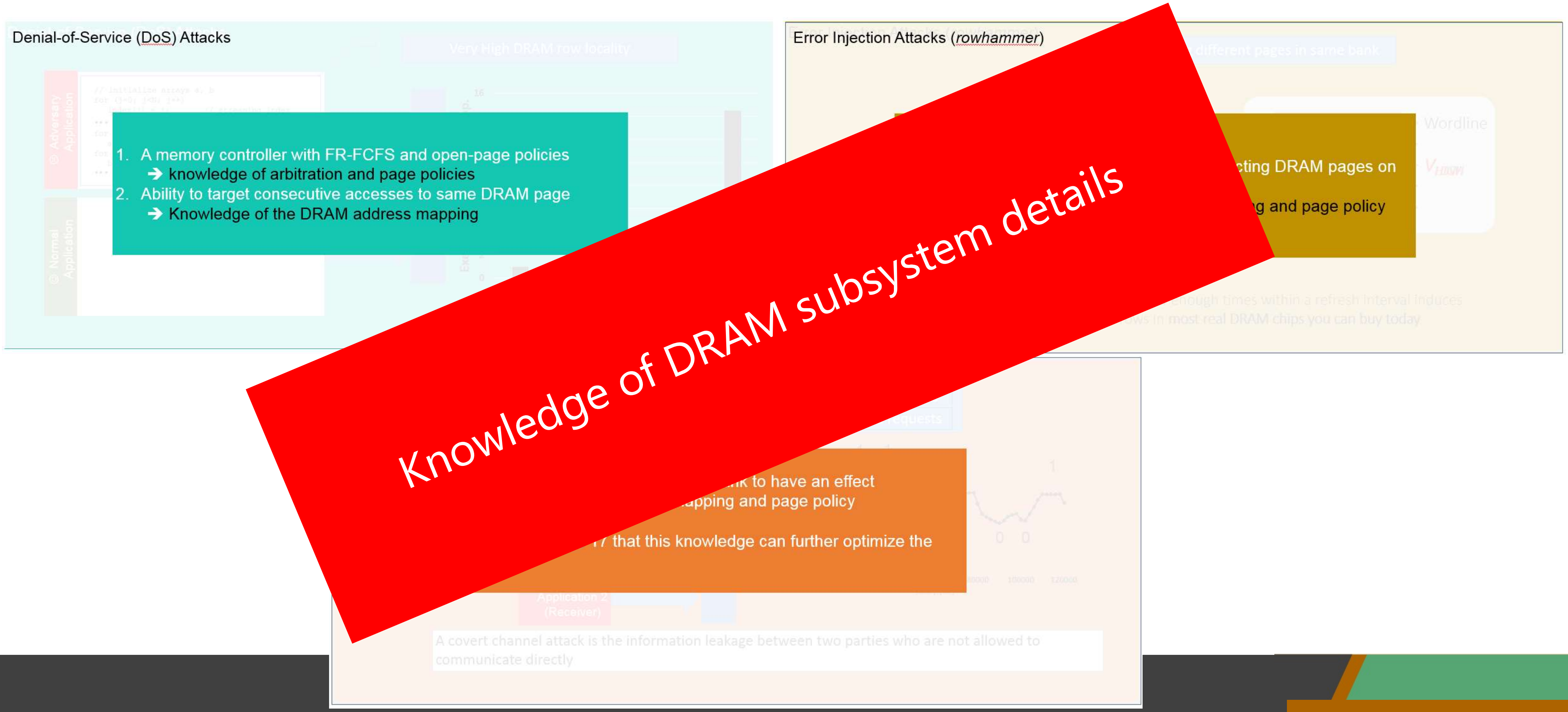
Targeting different pages in same bank



Ability to target consecutive accesses to conflicting DRAM pages on same bank

➔ Knowledge of the DRAM address mapping and page policy

Repeatedly opening and closing a row enough times within a refresh interval induces disturbance errors in adjacent rows in most real DRAM chips you can buy today



Denial-of-Service (DoS) Attacks

Very High DRAM row locality

1. A memory controller with FR-FCFS and open-page policies
   ➔ knowledge of arbitration and page policies
2. Ability to target consecutive accesses to same DRAM page
   ➔ Knowledge of the DRAM address mapping

Error Injection Attacks (*rowhammer*)

different pages in same bank

cting DRAM pages on

g and page policy

enough times within a refresh interval induces

ws in most real DRAM chips you can buy today

requests

nk to have an effect

apping and page policy

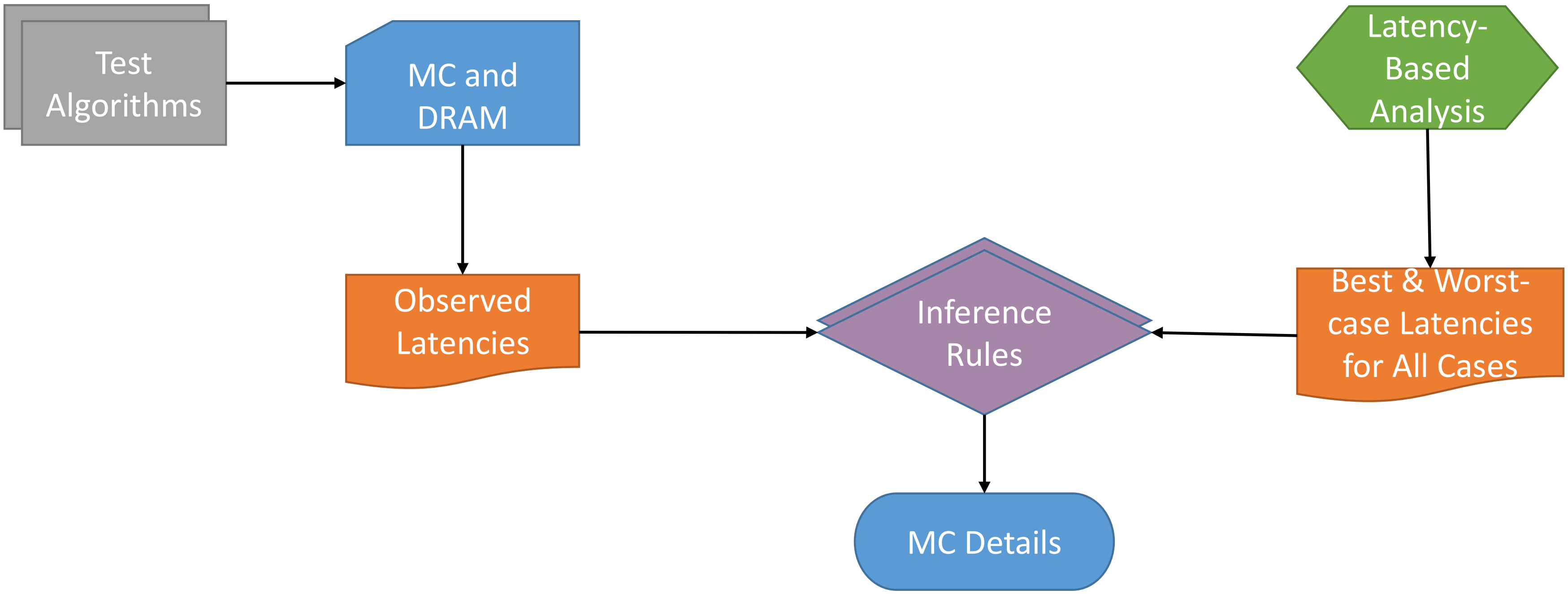
7 that this knowledge can further optimize the

A covert channel attack is the information leakage between two parties who are not allowed to communicate directly

Knowledge of DRAM subsystem details

MCReverse

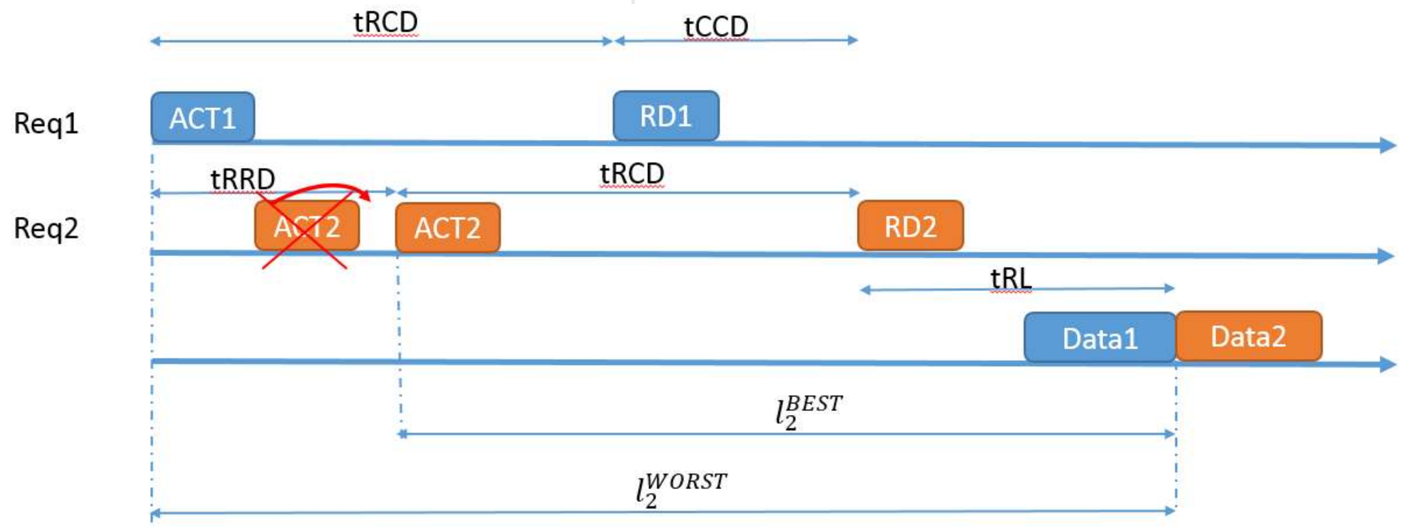
Security

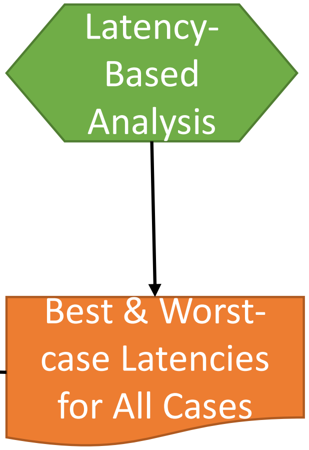


MCReverse

Security

Ex: Two read accesses to different banks same rank



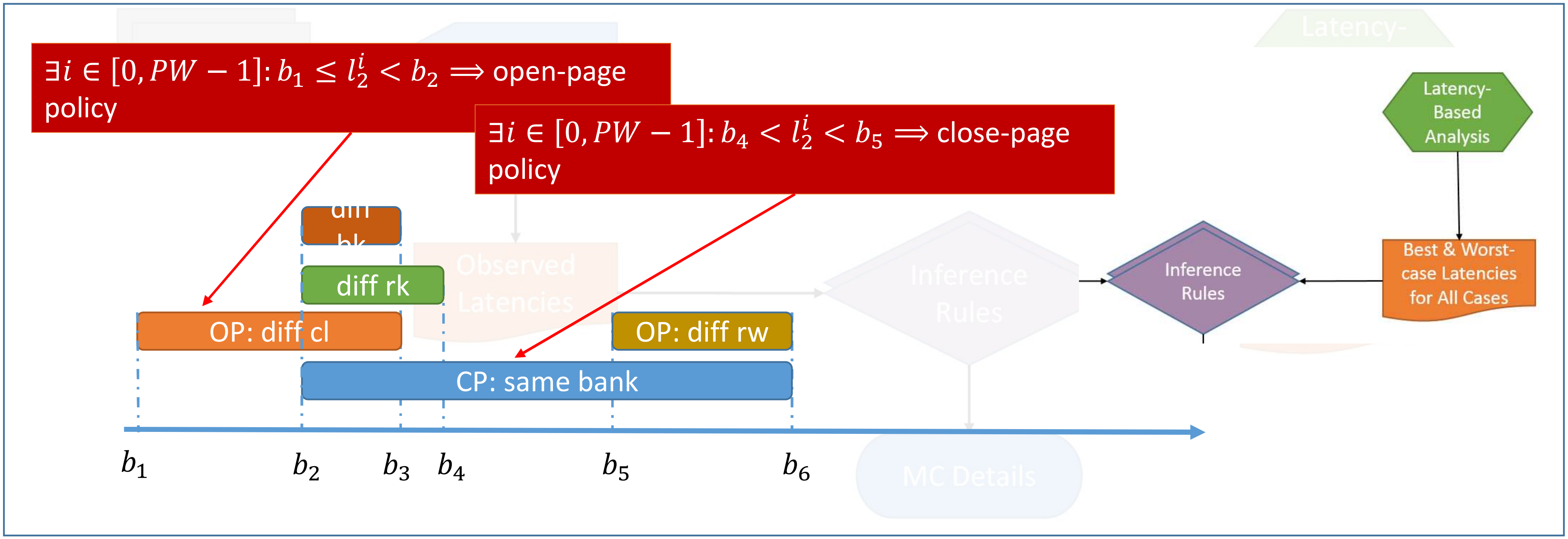


Latency-Based Analysis

Best & Worst-case Latencies for All Cases

MCReverse

Security

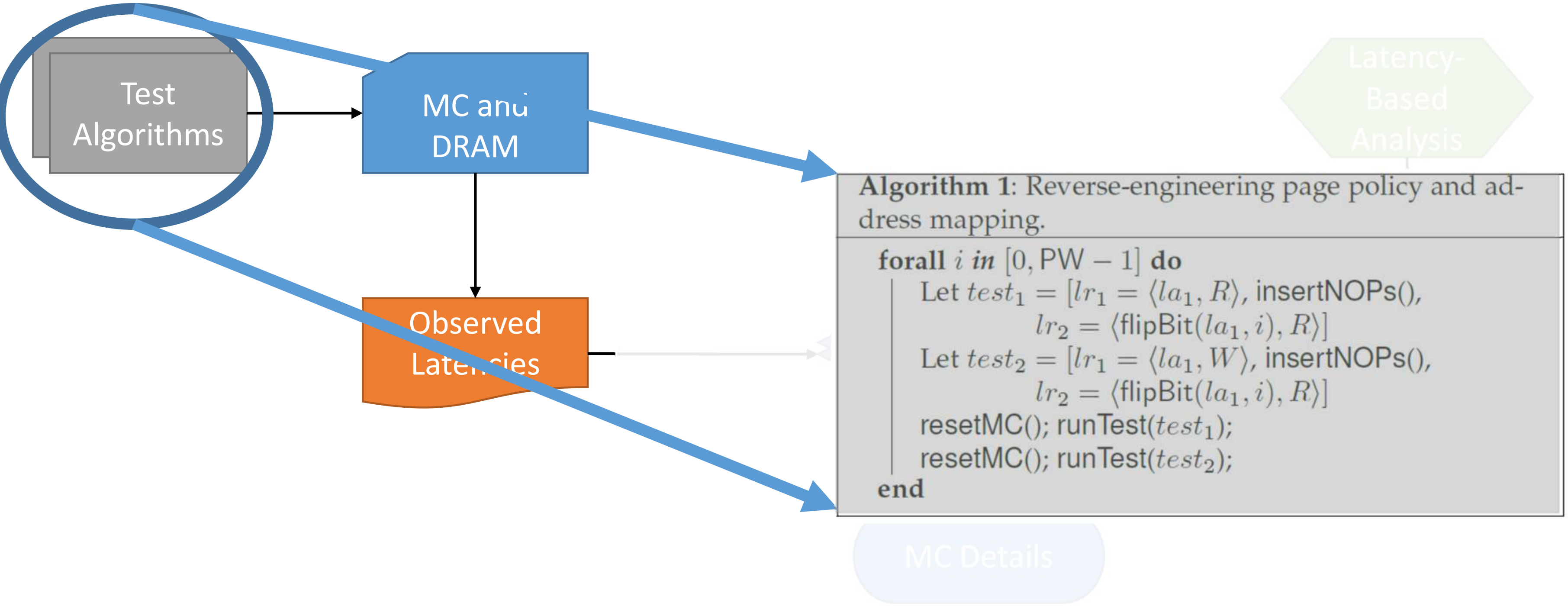


$\exists i \in [0, PW - 1]: b_1 \le l_2^i < b_2 \Longrightarrow$ open-page policy

$\exists i \in [0, PW - 1]: b_4 < l_2^i < b_5 \Longrightarrow$ close-page policy

MCReverse

Security

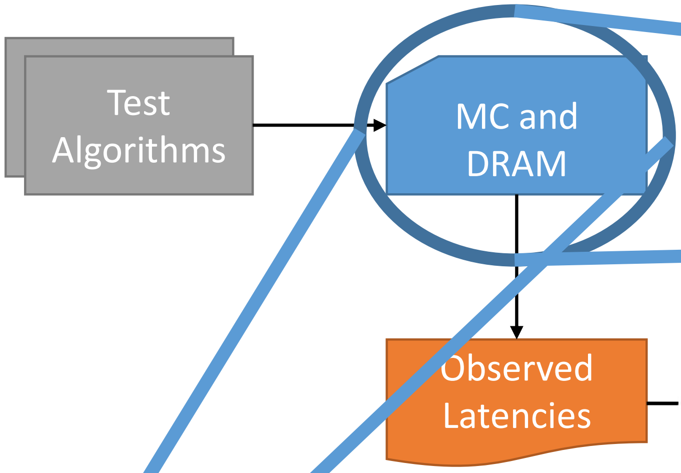


Algorithm 1: Reverse-engineering page policy and address mapping.

```
forall $i$ in $[0, \mathrm{PW}-1]$ do
    Let $test_1 = [lr_1 = \langle la_1, R\rangle$, insertNOPs(),
            $lr_2 = \langle$flipBit$(la_1, i), R\rangle]$
    Let $test_2 = [lr_1 = \langle la_1, W\rangle$, insertNOPs(),
            $lr_2 = \langle$flipBit$(la_1, i), R\rangle]$
    resetMC(); runTest($test_1$);
    resetMC(); runTest($test_2$);
end
```

MCReverse

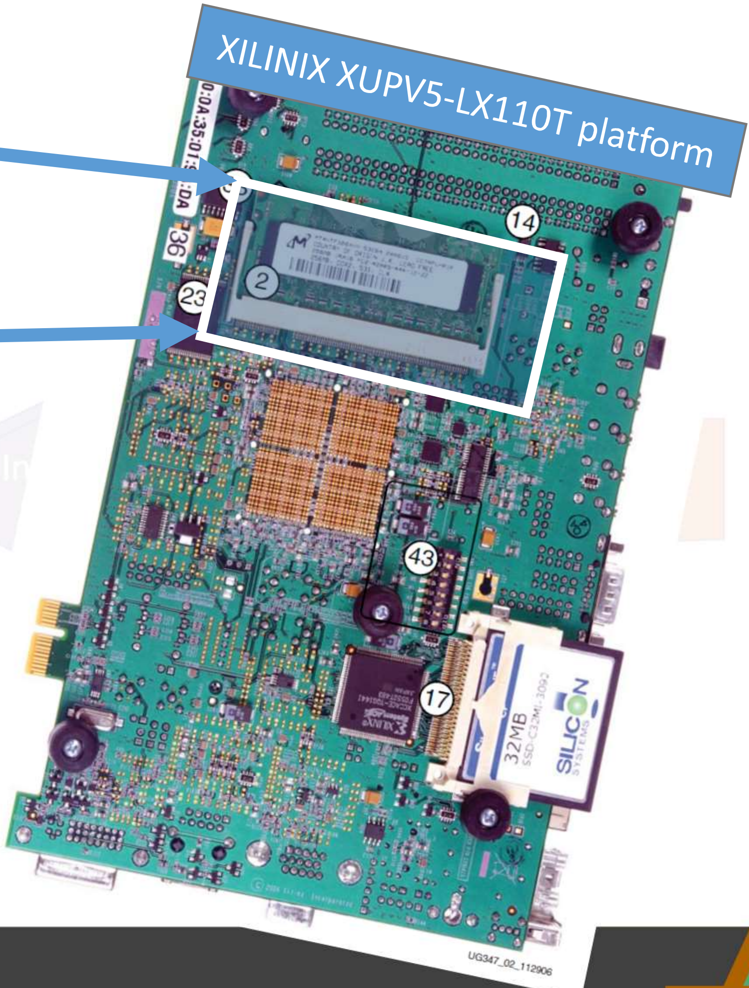
Security



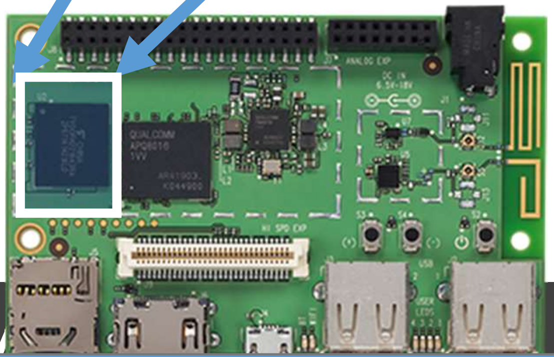
Test Algorithms

MC and DRAM

Observed Latencies



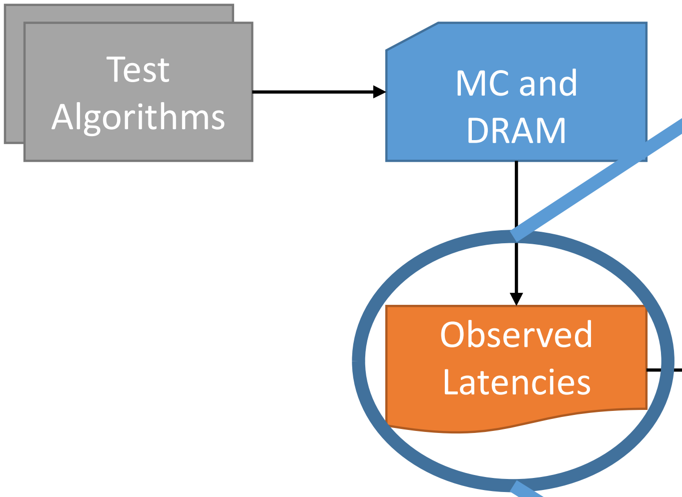
XILINIX XUPV5-LX110T platform

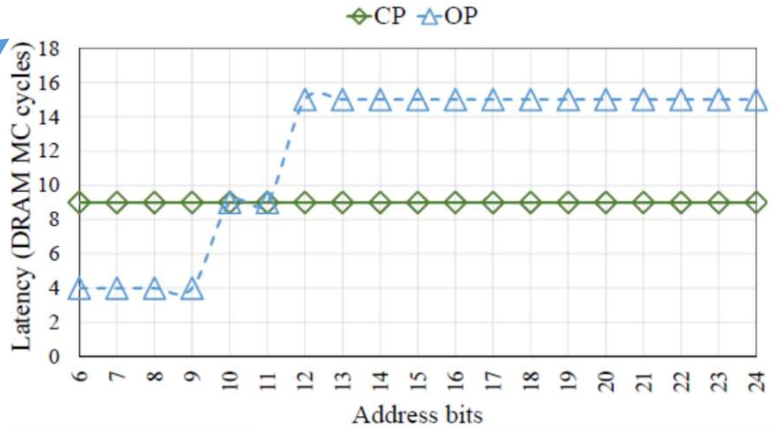


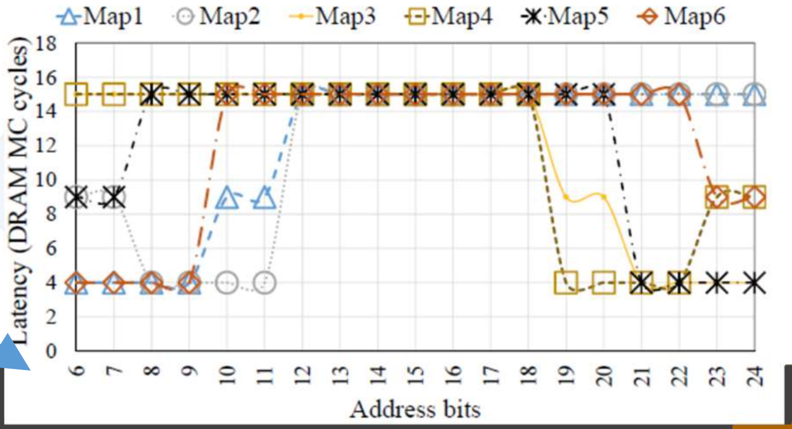
Qualcomm Dragon 410c IoT board

M

Security

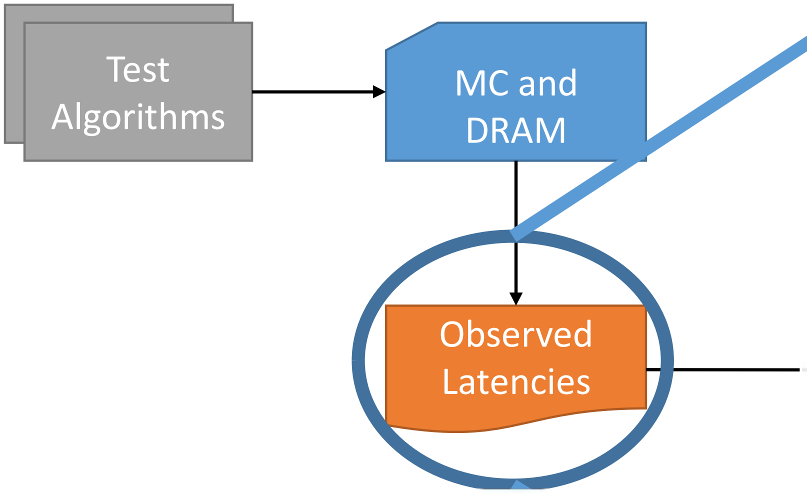






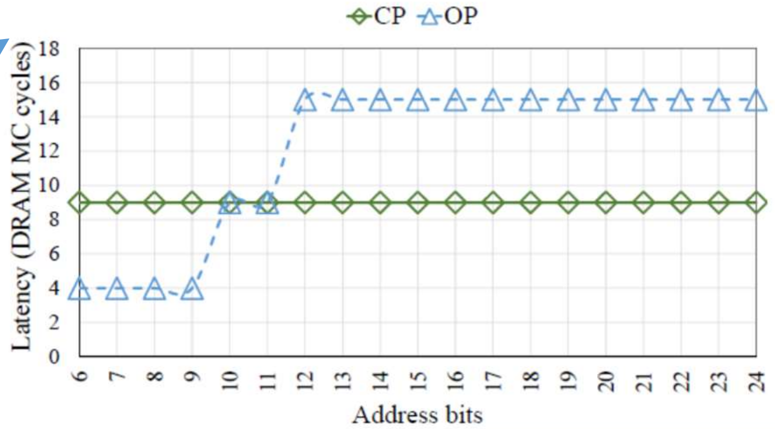
MCReverse

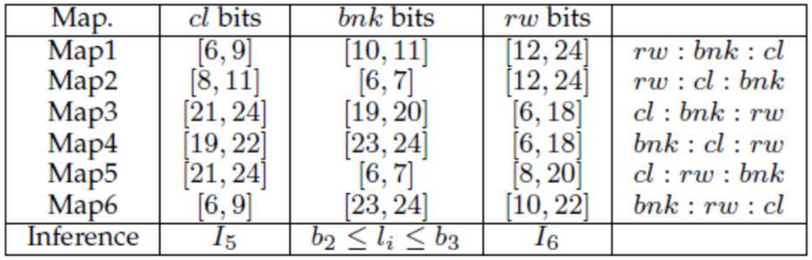
Security



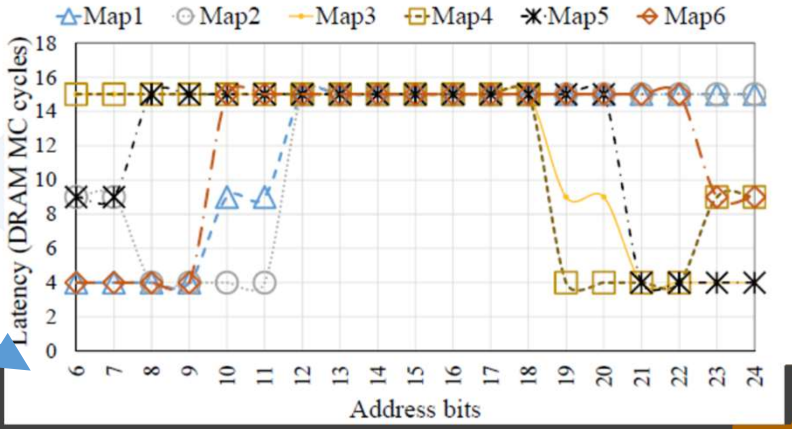
Inference Rules

case Latencies



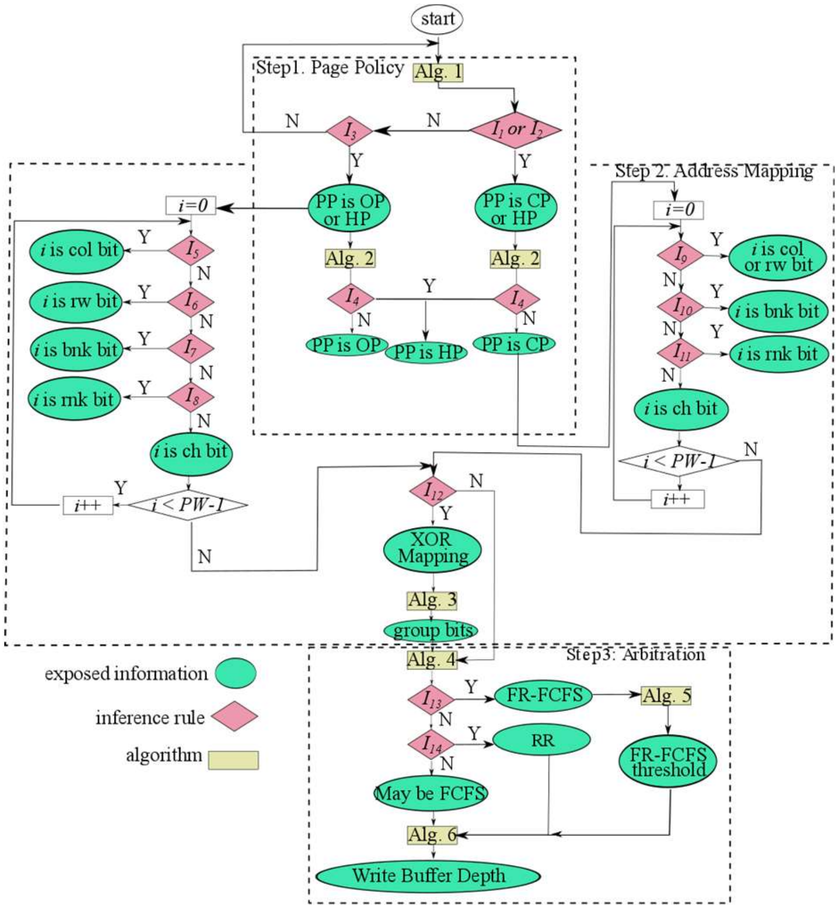


Map.	cl bits	bnk bits	rw bits	
Map1	[6, 9]	[10, 11]	[12, 24]	$rw : bnk : cl$
Map2	[8, 11]	[6, 7]	[12, 24]	$rw : cl : bnk$
Map3	[21, 24]	[19, 20]	[6, 18]	$cl : bnk : rw$
Map4	[19, 22]	[23, 24]	[6, 18]	$bnk : cl : rw$
Map5	[21, 24]	[6, 7]	[8, 20]	$cl : rw : bnk$
Map6	[6, 9]	[23, 24]	[10, 22]	$bnk : rw : cl$
Inference	I_5	$b_2 \leq l_i \leq b_3$	I_6	



MCReverse

Security



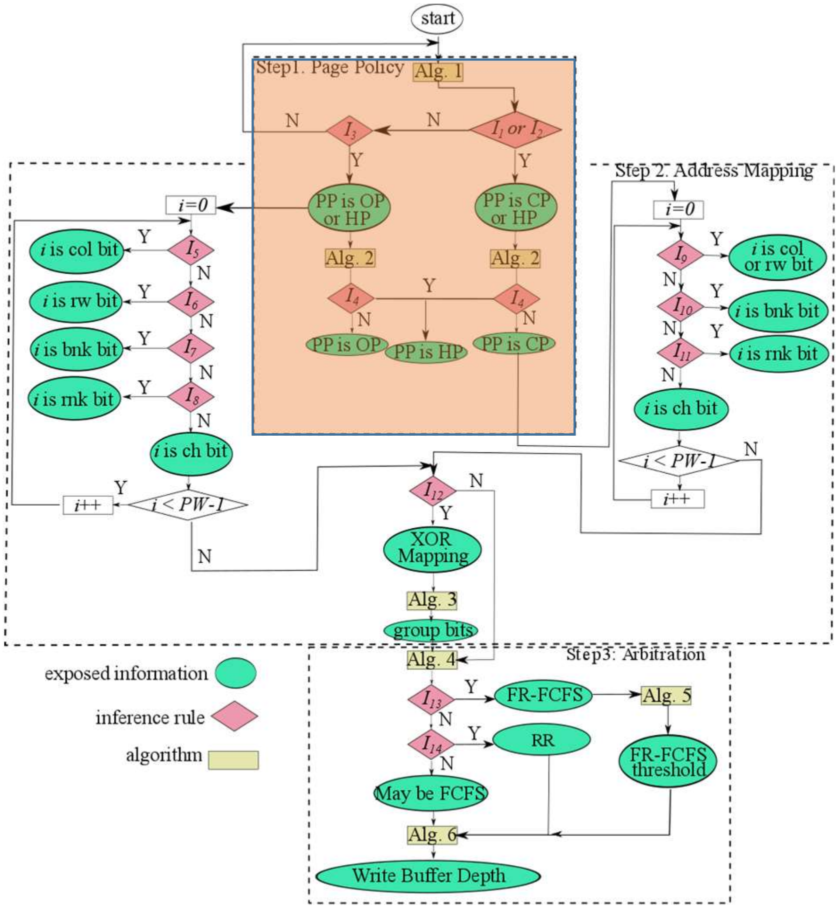
A Step-by-Step Process

SECURITY

A Step-by-Step Process

SECURITY

Step 1: Page policy
- ✓ Close-page
- ✓ Open-page
- ✓ Hybrid-page

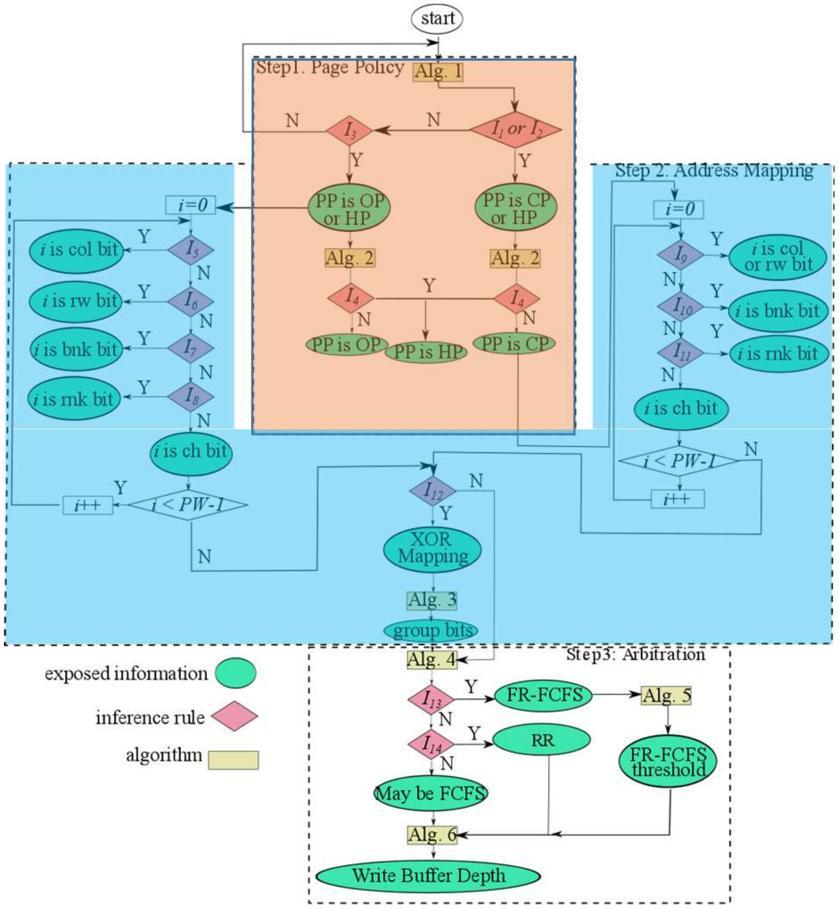


Step 1: Page policy

- ✓ Close-page
- ✓ Open-page
- ✓ Hybrid-page

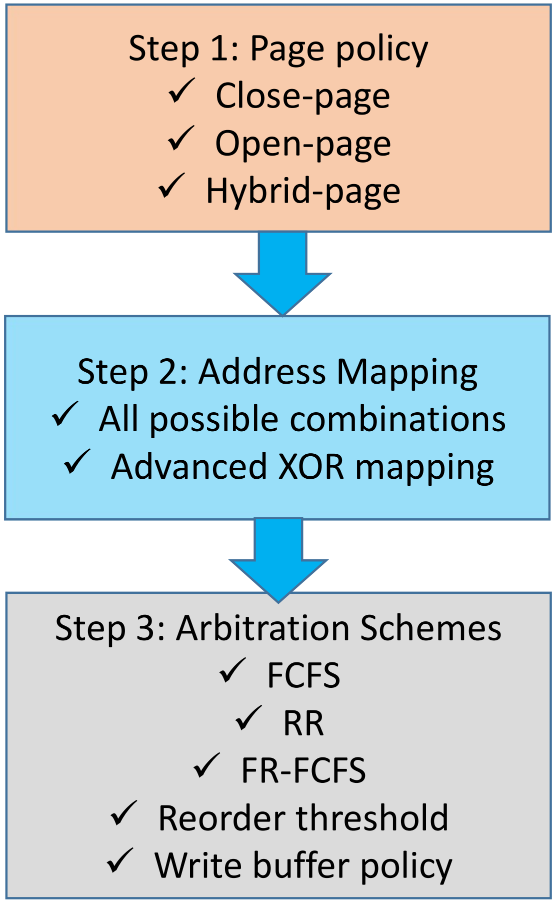
Step 2: Address Mapping

- ✓ All possible combinations
- ✓ Advanced XOR mapping



A Step-by-Step Process

SECURITY



Step 1: Page policy

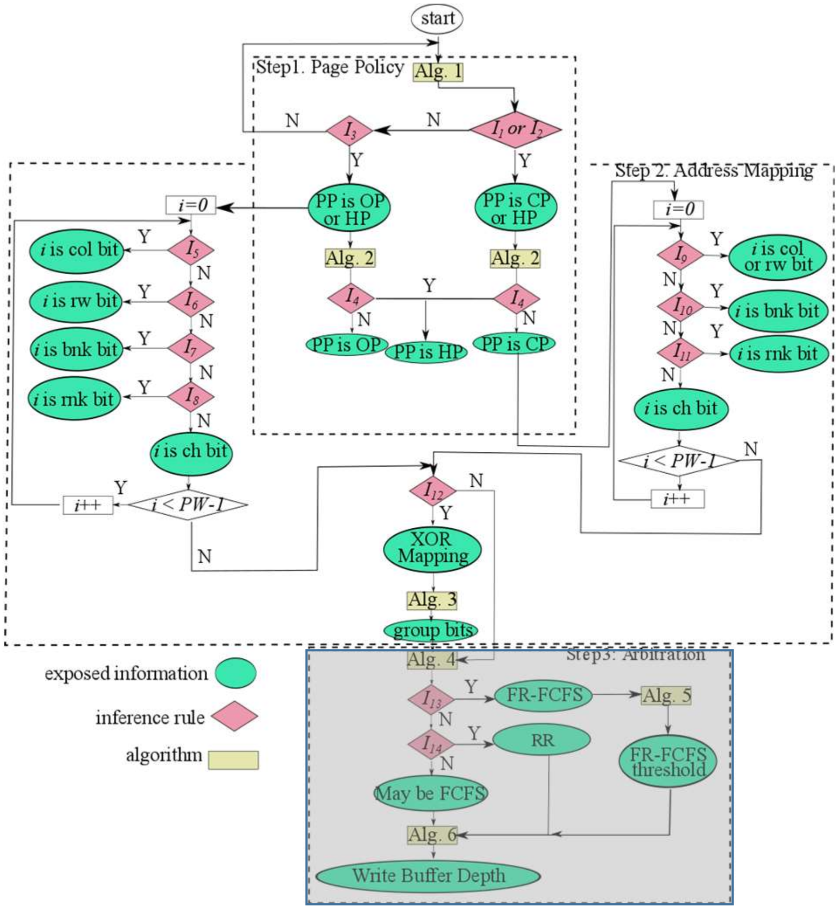
- ✓ Close-page
- ✓ Open-page
- ✓ Hybrid-page

Step 2: Address Mapping

- ✓ All possible combinations
- ✓ Advanced XOR mapping

Step 3: Arbitration Schemes

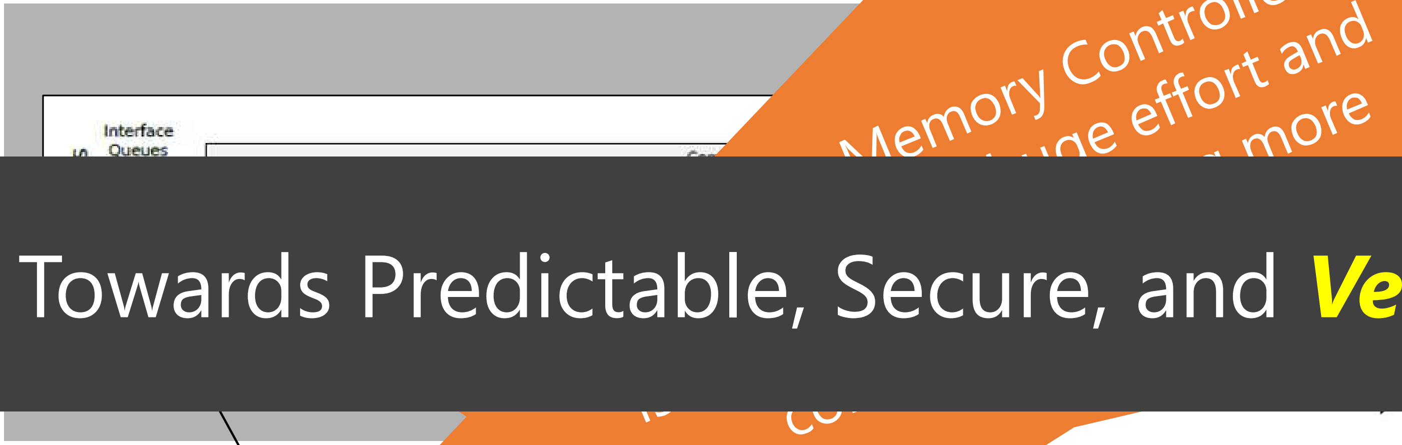
- ✓ FCFS
- ✓ RR
- ✓ FR-FCFS
- ✓ Reorder threshold
- ✓ Write buffer policy

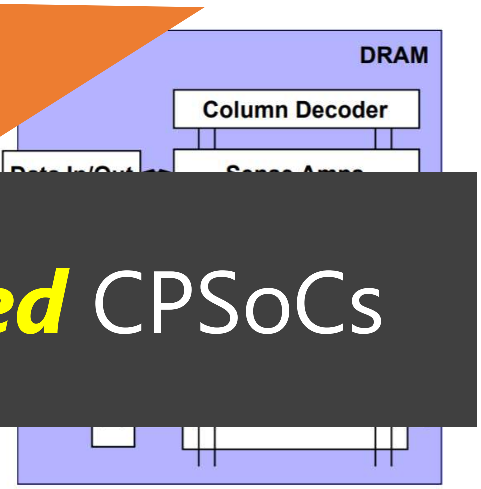


A Step-by-Step Process

SECURITY

Towards Predictable, Secure, and ***Verified*** CPSoCs





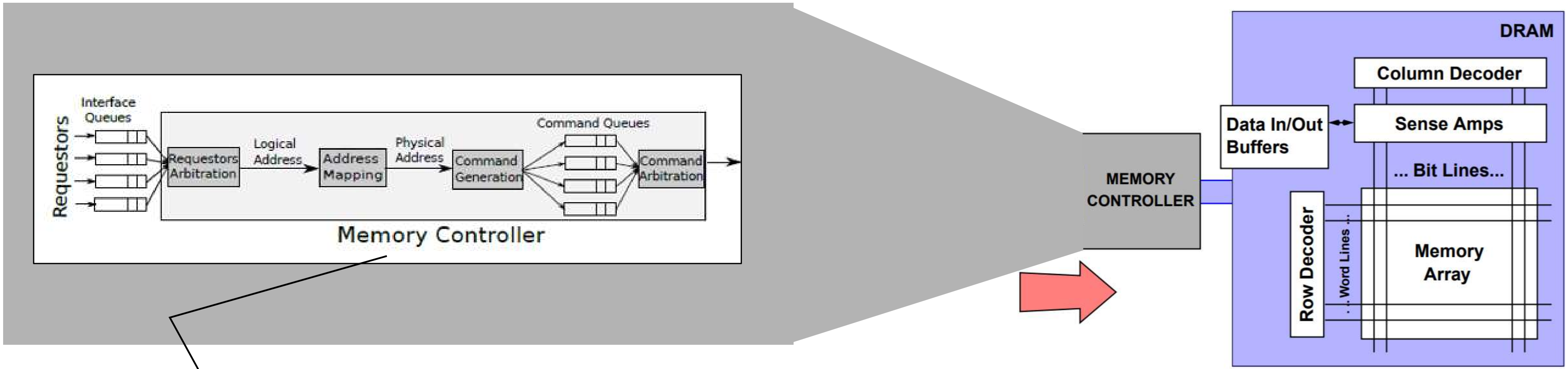
Memory Controller
...ge effort and
... more
... co...

✓ ...
Multiple reordering levels
✓ Various arbitration decisions

DRAM Systems are Complex

[DATE'16] ***Mohamed Hassan***, Hiren Patel, "MCXplore: An Automated Framework for Validating Memory Controller Designs"
[TCAD'17] ***Mohamed Hassan***, Hiren Patel, "MCXplore: Automating the Validation Process of DRAM Memory Controller Designs"

VERIFICATION

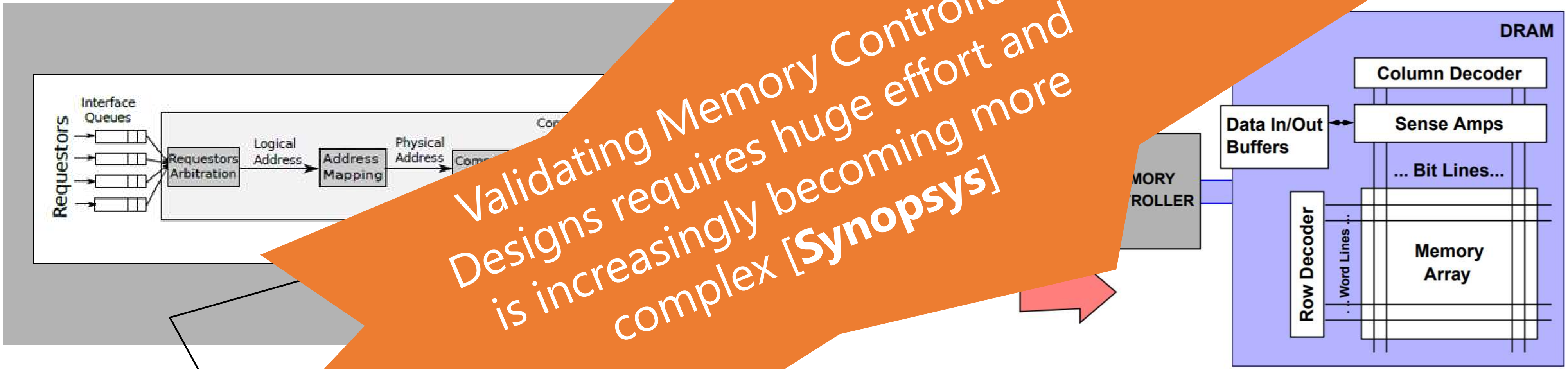


- ✓ Complex optimizations
- ✓ Multiple reordering levels
- ✓ Various arbitration decisions

DRAM Systems are Complex

*[DATE'16] **Mohamed Hassan**, Hiren Patel, "MCXplore: An Automated Framework for Validating Memory Controller Designs"*
*[TCAD'17] **Mohamed Hassan**, Hiren Patel, "MCXplore: Automating the Validation Process of DRAM Memory Controller Designs"*

VERIFICATION



Validating Memory Controller Designs requires huge effort and is increasingly becoming more complex [**Synopsys**]

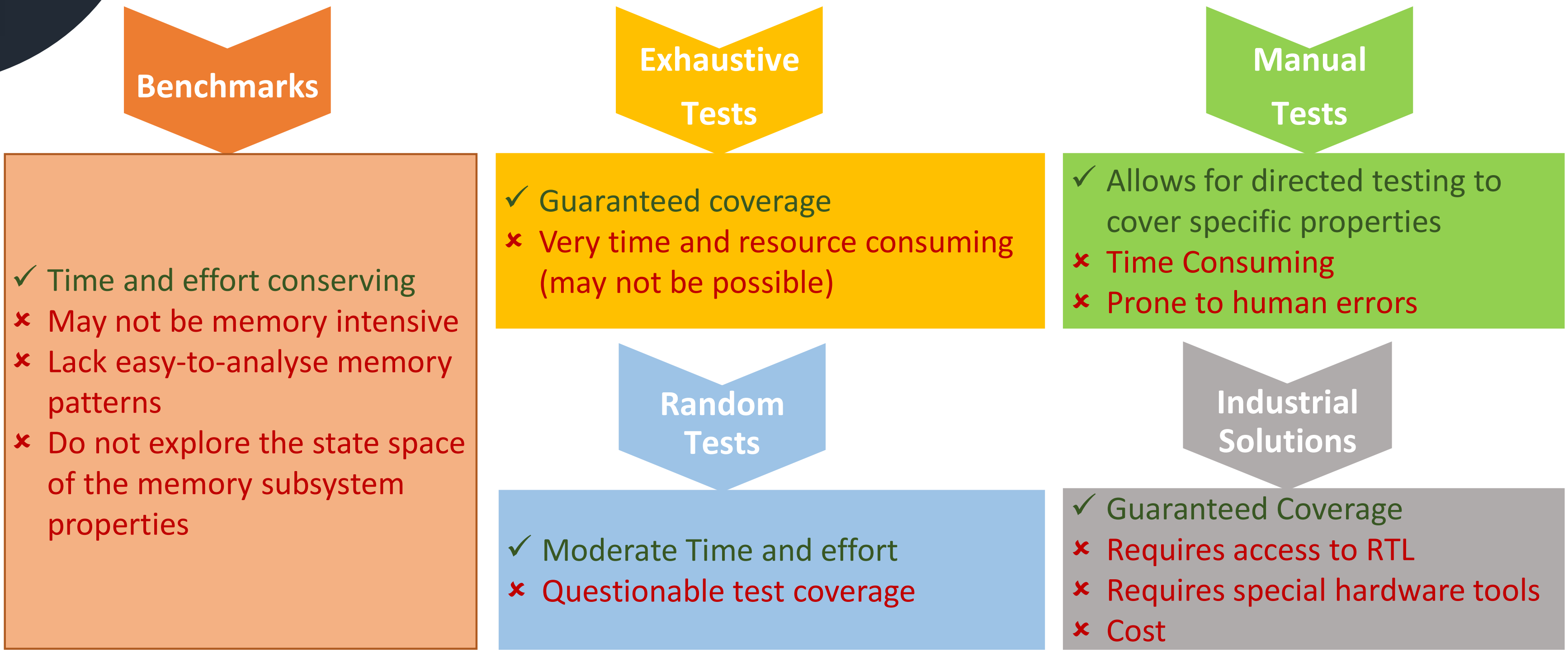
- ✓ Multiple reordering levels
- ✓ Various arbitration decisions

DRAM Systems are Complex

*[DATE'16] **Mohamed Hassan**, Hiren Patel, "MCXplore: An Automated Framework for Validating Memory Controller Designs"*

*[TCAD'17] **Mohamed Hassan**, Hiren Patel, "MCXplore: Automating the Validation Process of DRAM Memory Controller Designs"*

VERIFICATION



Benchmarks

- ✓ Time and effort conserving
- × May not be memory intensive
- × Lack easy-to-analyse memory patterns
- × Do not explore the state space of the memory subsystem properties

Exhaustive Tests

- ✓ Guaranteed coverage
- × Very time and resource consuming (may not be possible)

Manual Tests

- ✓ Allows for directed testing to cover specific properties
- × Time Consuming
- × Prone to human errors

Random Tests

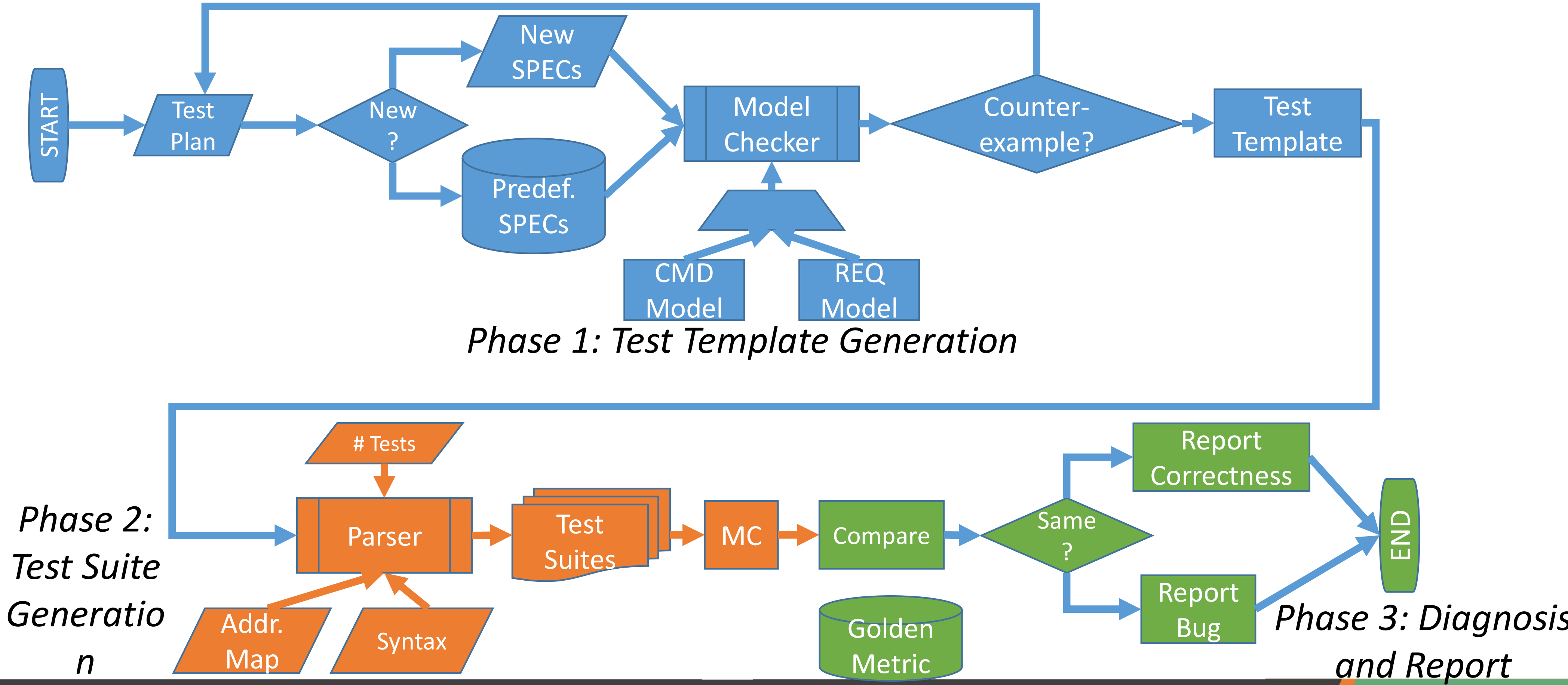
- ✓ Moderate Time and effort
- × Questionable test coverage

Industrial Solutions

- ✓ Guaranteed Coverage
- × Requires access to RTL
- × Requires special hardware tools
- × Cost

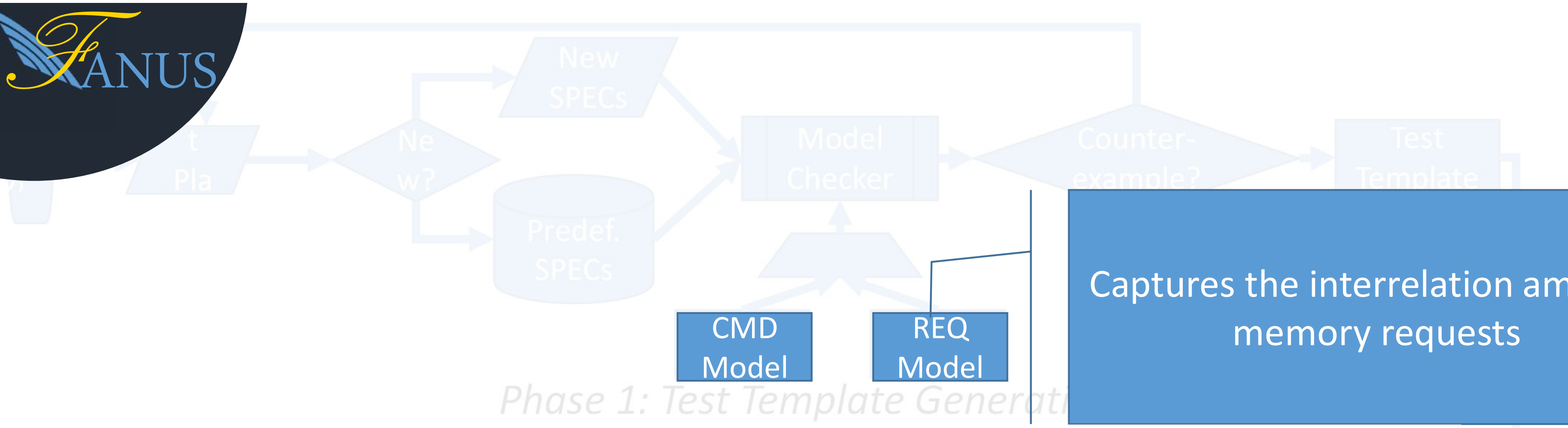
Existing Solutions

VERIFICATION

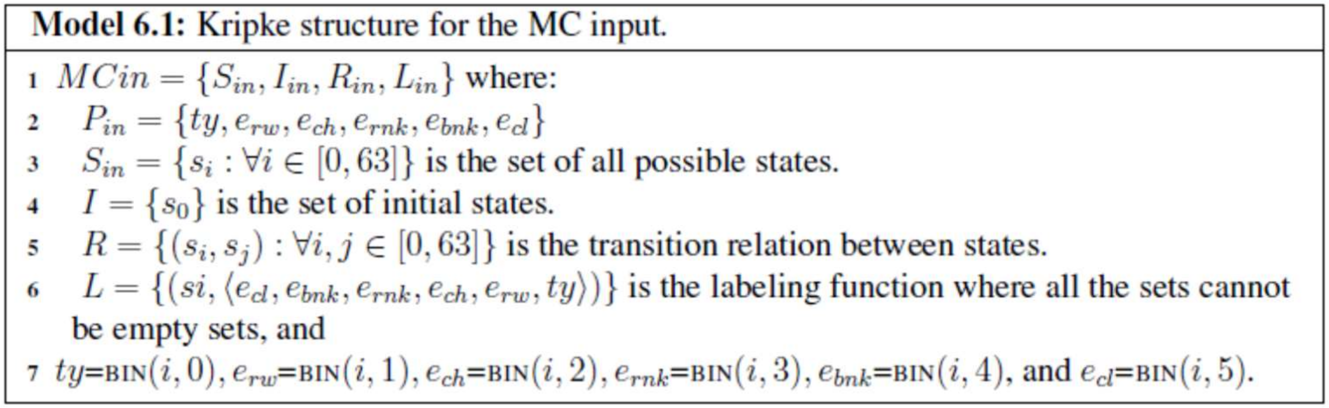


MCXplore: Big Picture

VERIFICATION



Captures the interrelation amongst memory requests



Model 6.1: Kripke structure for the MC input.

1 $MCin = \{S_{in}, I_{in}, R_{in}, L_{in}\}$ where:
2 $P_{in} = \{ty, e_{rw}, e_{ch}, e_{rnk}, e_{bnk}, e_{cl}\}$
3 $S_{in} = \{s_i : \forall i \in [0, 63]\}$ is the set of all possible states.
4 $I = \{s_0\}$ is the set of initial states.
5 $R = \{(s_i, s_j) : \forall i, j \in [0, 63]\}$ is the transition relation between states.
6 $L = \{(si, \langle e_{cl}, e_{bnk}, e_{rnk}, e_{ch}, e_{rw}, ty\rangle)\}$ is the labeling function where all the sets cannot be empty sets, and
7 ty=BIN$(i, 0)$, e_{rw}=BIN$(i, 1)$, e_{ch}=BIN$(i, 2)$, e_{rnk}=BIN$(i, 3)$, e_{bnk}=BIN$(i, 4)$, and e_{cl}=BIN$(i, 5)$.

Phase 1: Test Template Generation

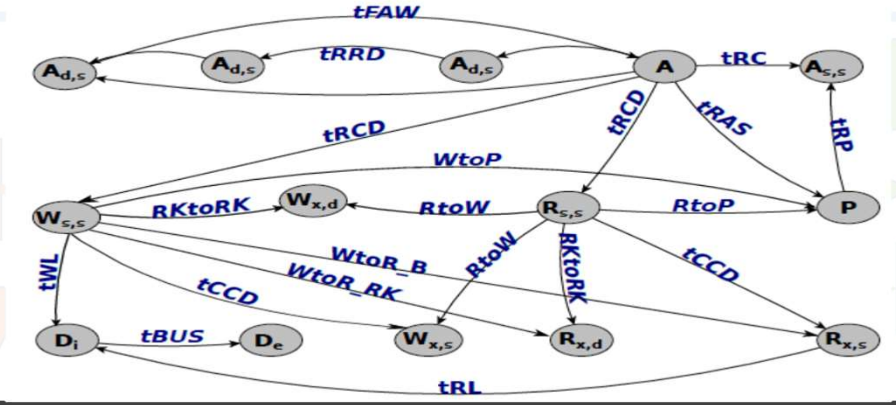
VERIFICATION

Interactions between memory commands and their timing constraints

CMD Model

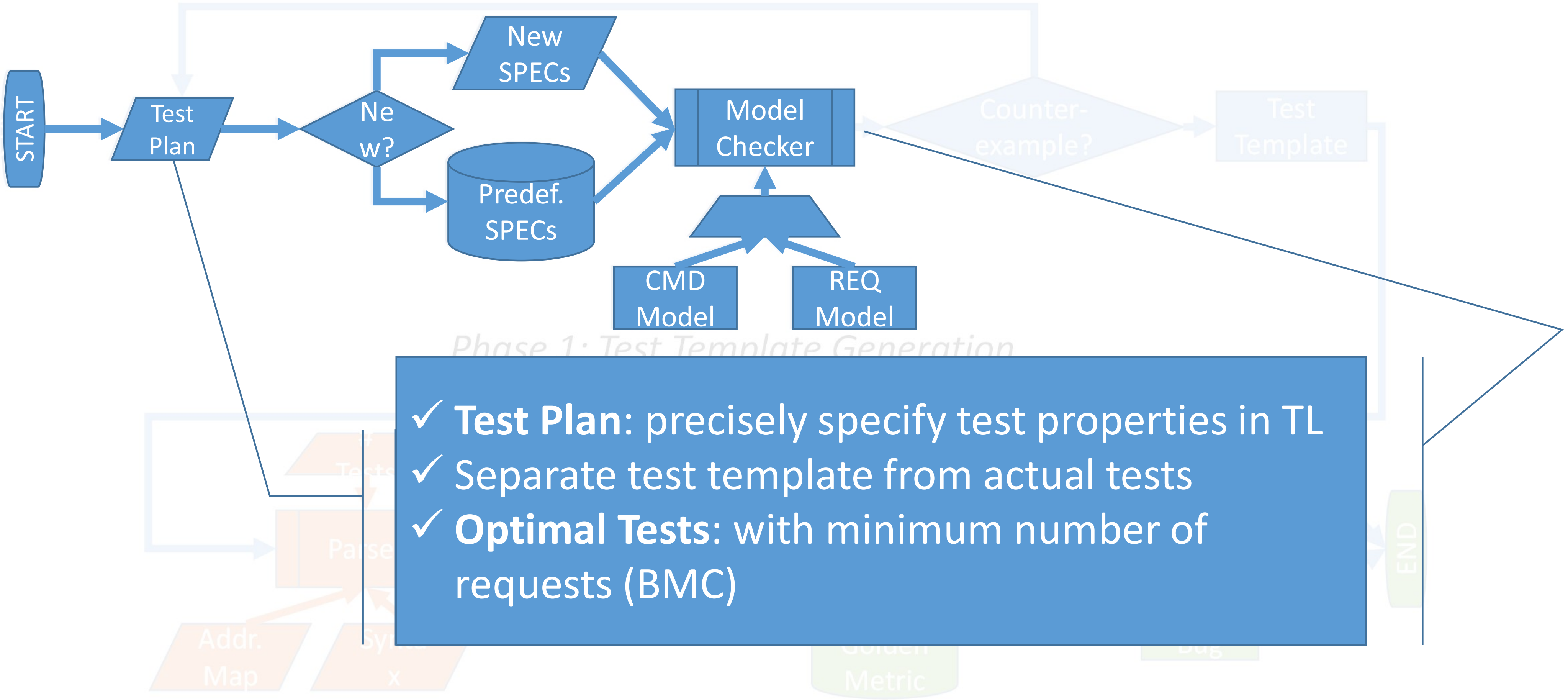
REQ Model

Captures the interrelation amongst memory requests



Phase 1: Test Template Generation

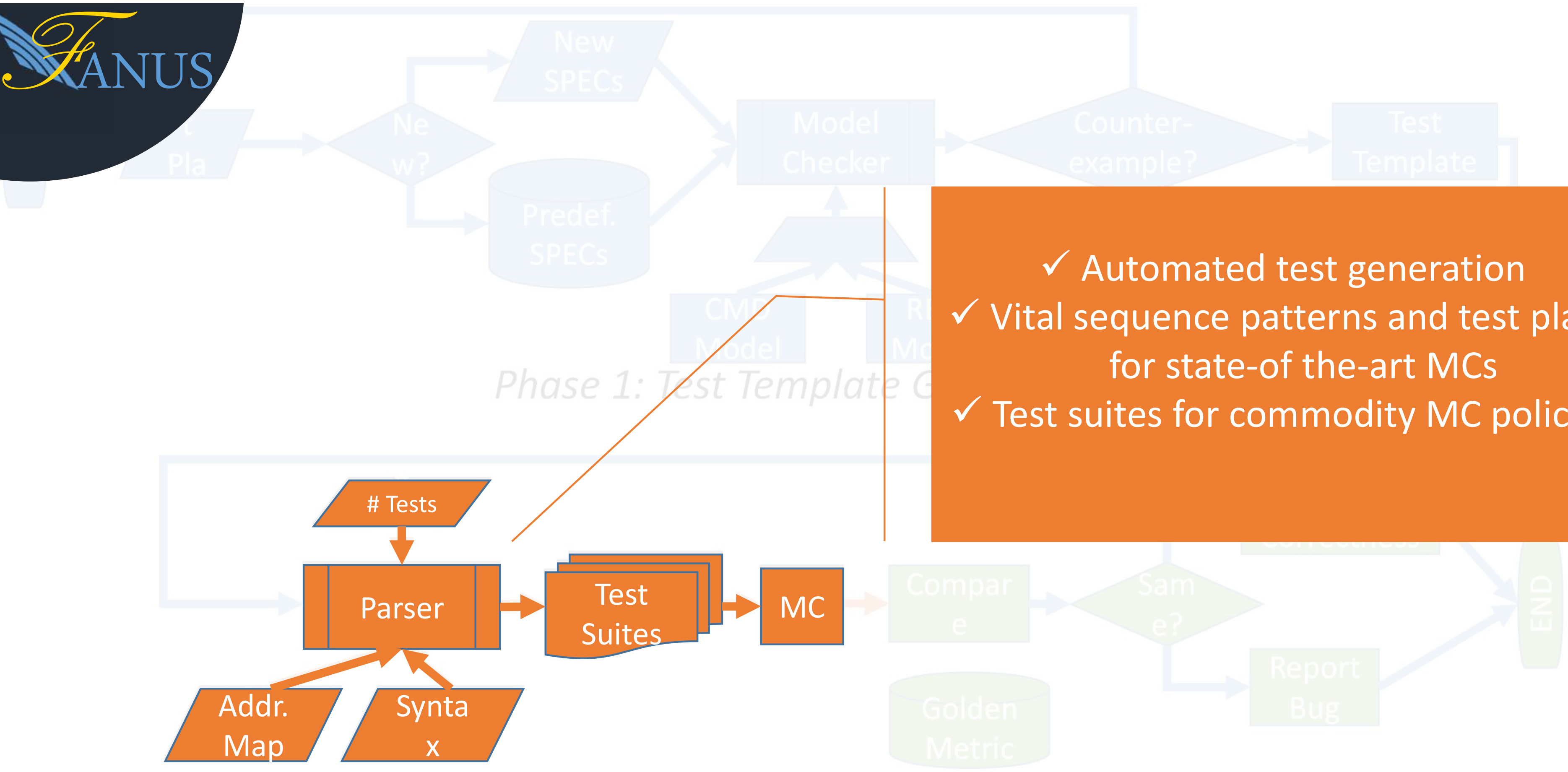
VERIFICATION



- ✓ **Test Plan**: precisely specify test properties in TL
- ✓ Separate test template from actual tests
- ✓ **Optimal Tests**: with minimum number of requests (BMC)

Phase 1: Test Template Generation

VERIFICATION

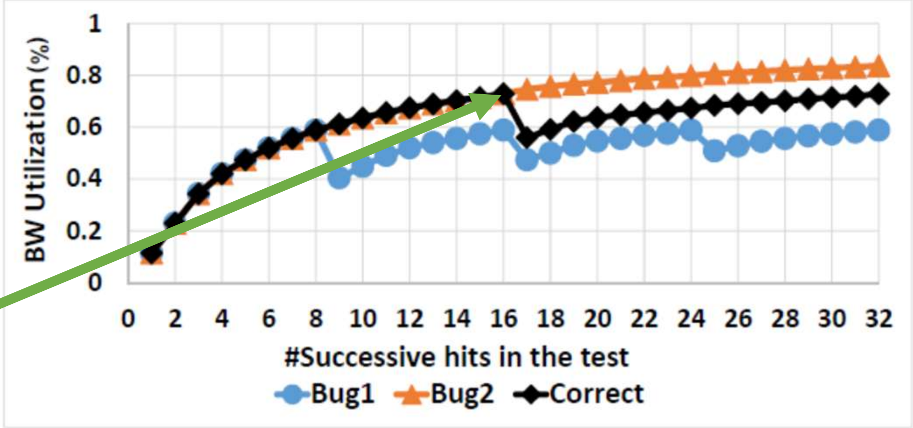


✓ Automated test generation
✓ Vital sequence patterns and test plans for state-of the-art MCs
✓ Test suites for commodity MC policies

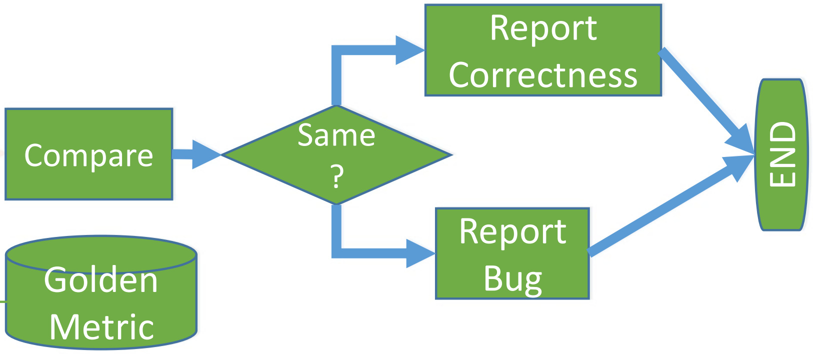
Phase 2: Test Suite Generation

VERIFICATION

- ✓ High-level statistics such as bandwidth
  - ✓ do not require internal debugging capabilities (*black box technique*)



$$U_c = \frac{thr \times tBUS}{tRCD + (thr - 1)tCCD + RtoP + tRP} \cong 73\%$$



Phase 3: Diagnosis and Report

VERIFICATION

CPSoCs Opportunities for MCS

1. MPSoCs create switching alternatives

- Different modes of operation at different cluster of PEs?

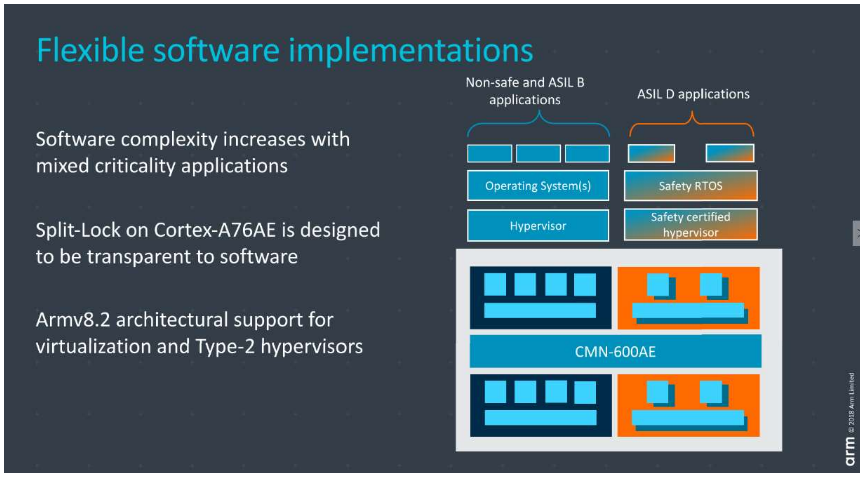
Back to the Big Picture

Future Directions

CPSoCs Opportunities for MCS

1. MPSoCs create switching alternatives

- Different modes of operation at different cluster of PEs?



Flexible software implementations

Software complexity increases with mixed criticality applications

Split-Lock on Cortex-A76AE is designed to be transparent to software

Armv8.2 architectural support for virtualization and Type-2 hypervisors

Back to the Big Picture

Future Directions

CPSoCs Opportunities for MCS

1. MPSoCs create switching alternatives

- Different modes of operation at different cluster of PEs?
- Migrate instead of switching?
  - Dynamic Reconfiguration (IEC61508-7)

> ***C.3.13 Dynamic reconfiguration***
>
> ***The logical architecture of the system has to be such that it can be mapped onto a subset of the available resources of the system. The architecture needs to be capable of detecting a failure in a physical resource and then remapping the logical architecture back onto the restricted resources left functioning. Although the concept is more traditionally restricted to recovery from failed hardware units, it is also applicable to failed software units if there is sufficient 'run-time redundancy' to allow a software re-try or if there is sufficient redundant data to make the individual and isolated failure be of little importance. This technique must be considered at the first system design stage.***

FANUS

CPSoCs Opportunities for MCS

. MPSoCs create switching alternatives

- Different modes of operation at different cluster of PEs?
- Migrate instead of switching?

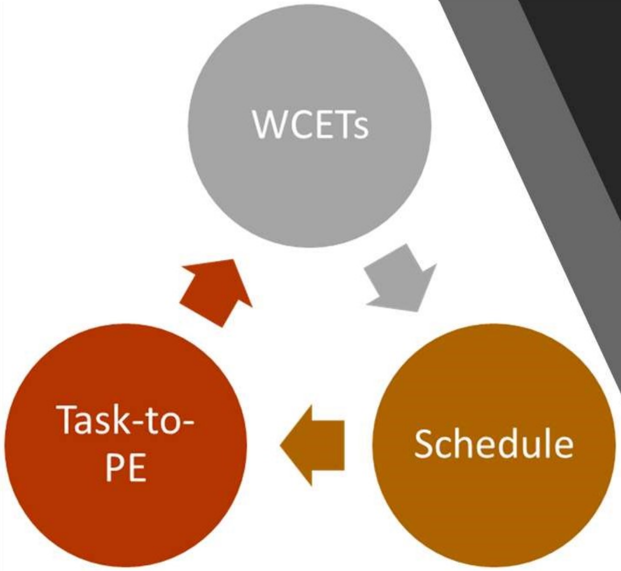
2. MPSoCs open the door for customized solutions

- Using specialized PEs is a norm in MPSoCs
- Dedicating a PE for the runtime monitoring
  - faster detection of exceptional events → react in a timely manner
- PE can be further tailored to optimize the behavior of the monitoring techniques

Back to the Big Picture

Future Directions

CPSoCs Challenges in MCS

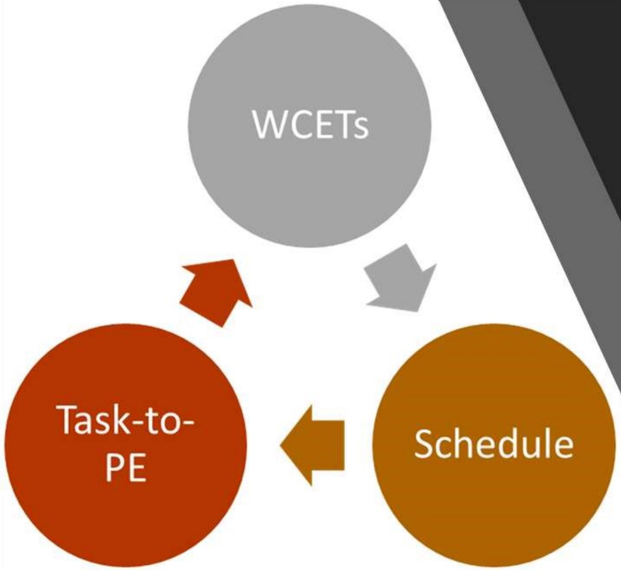


1. Common assumption:

   "uncertainty in WCET does not come from the system itself; rather, it comes from our inability to measure (or compute) it with complete confidence"

- Well, this may not be completely true for MPSoCs
  - ➢In SMPs, which core (or cores) executing a task does not affect its measured execution time.
  - ➢In MPSoCs, this decision directly affects the level of certainty in its WCET:
    - Real-time vs High-performance PEs?
    - Use scratchpads vs caches?

CPSoCs Challenges in MCS



2. Scalability challenges associated with these scheduling and monitoring techniques.

3. Mode switching in MPSoCs may incur task migrations or reassignment of heterogeneous cores to tasks

   - ➢ the effects of these decisions on the switching overhead need to be quantified.

CPSoCs Opportunities for Predictability

All about Flexibility

1. Which memory levels should be shared amongst which cores
   - Does the GPU share the LLC with the CPU?
2. How to distribute the cache architecture?
   - Would implementing a NUCA be adequate for MCS (e.g., helping in achieving different levels of isolation)?
3. Different types of on-chip memories
   - Both caches and SPMs
   - Most of the currently available approaches focus on a single type
4. Different types of available off-chip memories
   - DDR, GDDR, RLDRAM, LPDDR, QDR.
   - Investigating the cooperation of these types is also worth investigating

Back to the Big Picture

Future Directions

CPSoCs Challenges for Predictability

1. The interference exaggerates with the increase in the number of PEs
2. Understanding the architectural details of shared resources is inevitable to derive realistic bounds.
   - ***[MCS-MPSoCs, EMSOFT' 18]***
3. Each type of PEs has its own memory access behavior, which complicates the analysis, leading to more pessimism
   - Data-intensive PEs (e.g. multimedia/DSP processors) can saturate system queues
   - ***A requirement- and criticality-aware:***
     - ***Interconnect [CArb, RTAS'16]***
     - ***DRAM MC [PMC, RTAS'15&TECS'16]***

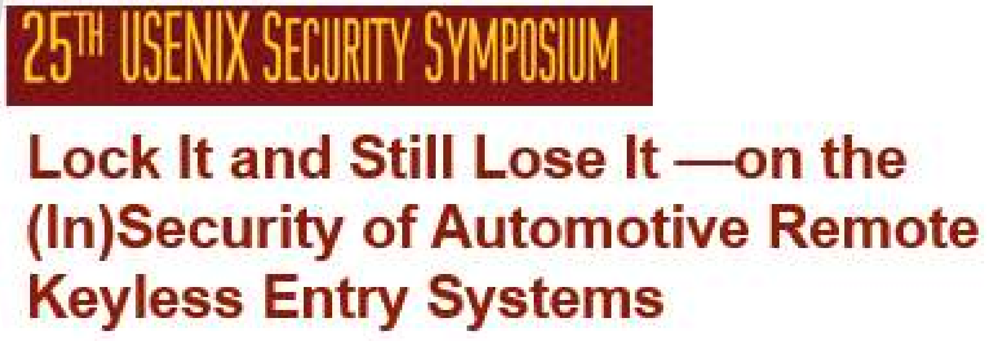
CPSoCs Challenges in Security

CYBER-PHYSICAL NATURE

HETEROGENEITY OF CPSOCS

SHARED COMPONENTS (AGAIN!)

Cyber-physical Nature



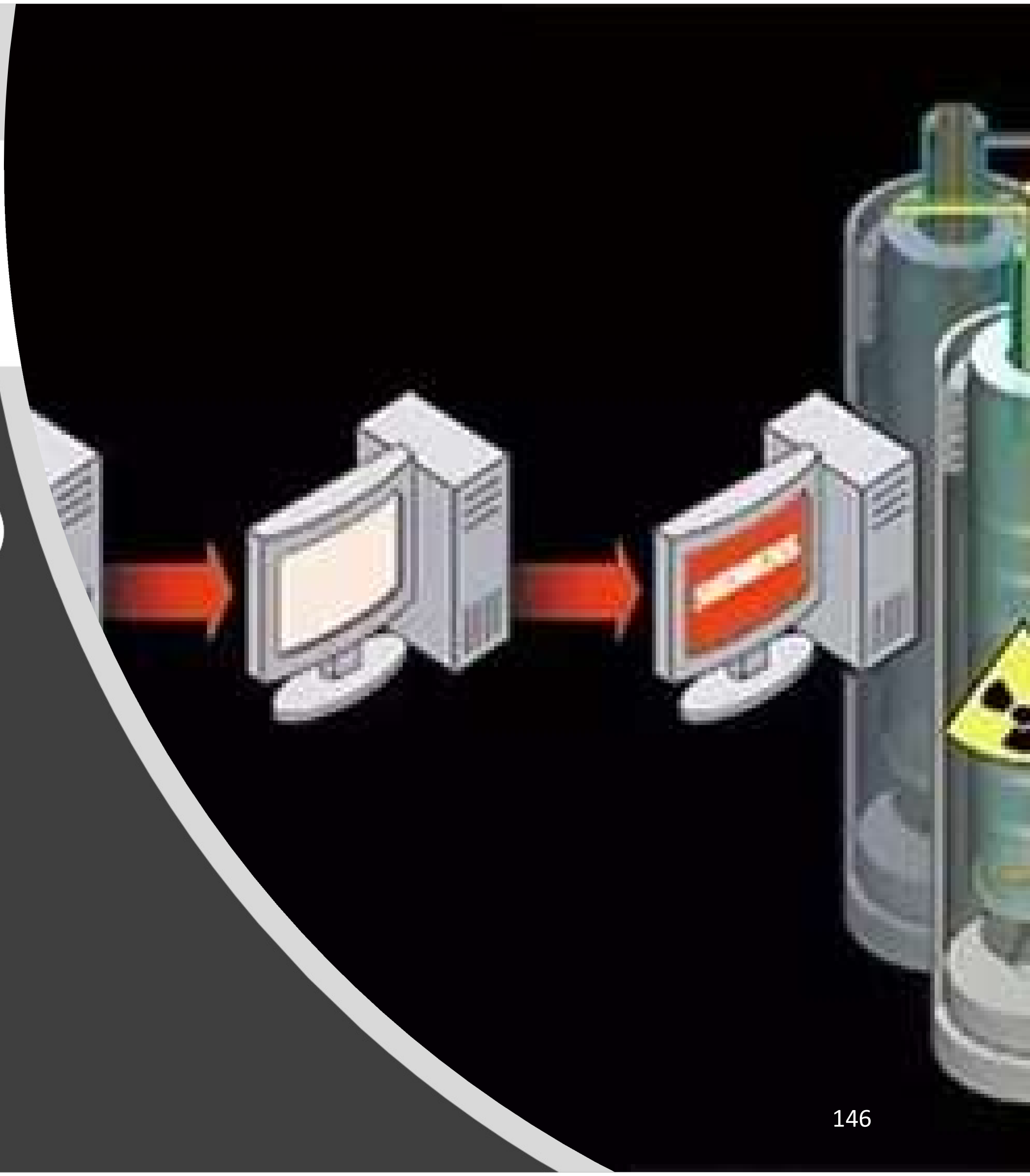
25TH USENIX SECURITY SYMPOSIUM

Lock It and Still Lose It —on the (In)Security of Automotive Remote Keyless Entry Systems

- CPS manage sensitive tasks in critical domains: power grids, cars, factories, nuclear plants
- Any security breach could lead to catastrophic consequences
- Hackers gained access to locked cars by only eavesdropping a single signal from the original remote keyless entry unit of the car

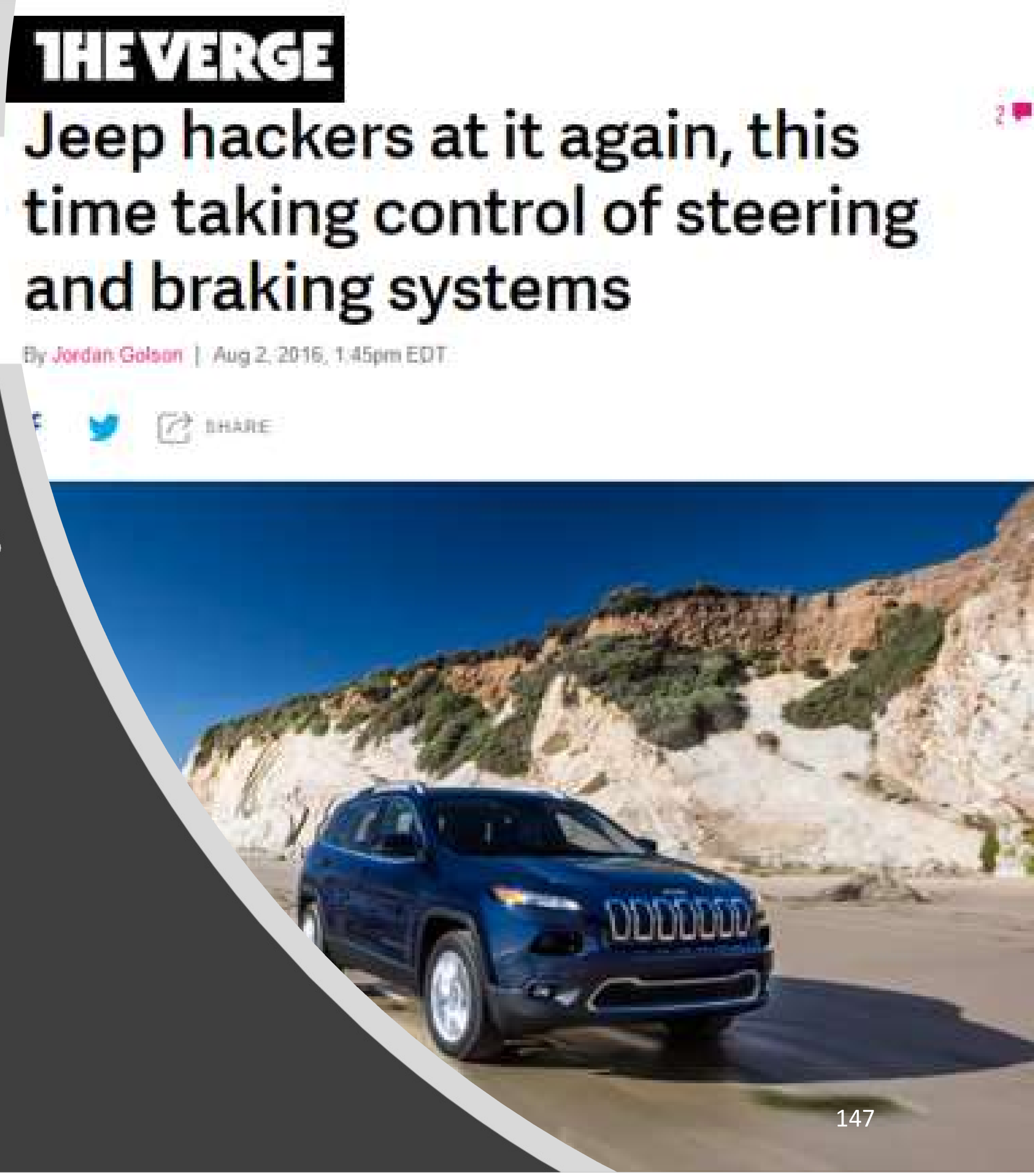
Heterogeneity of CPSoCs

- Each PE can be a 3rd-party IP (40% at Intel!)
- PEs share system components and interact with each other →new across-PEs threats
- Stuxnet attack exploited the authentication of the Siemens programmable logic controller by an access to a Windows machine



Shared hardware components in CPSoCs

- Historically, security was not considered as a concern for CPS because of isolation
- Not the case anymore
- Researchers were able to control sensitive (considered secure) engine control by compromising the (considered insecure) radio unit
  - Reason? Sharing the CAN



THE VERGE

Jeep hackers at it again, this time taking control of steering and braking systems

By Jordan Golson | Aug 2, 2016, 1:45pm EDT

SHARE

FANUS

Possible Directions for Security in CPSoCs

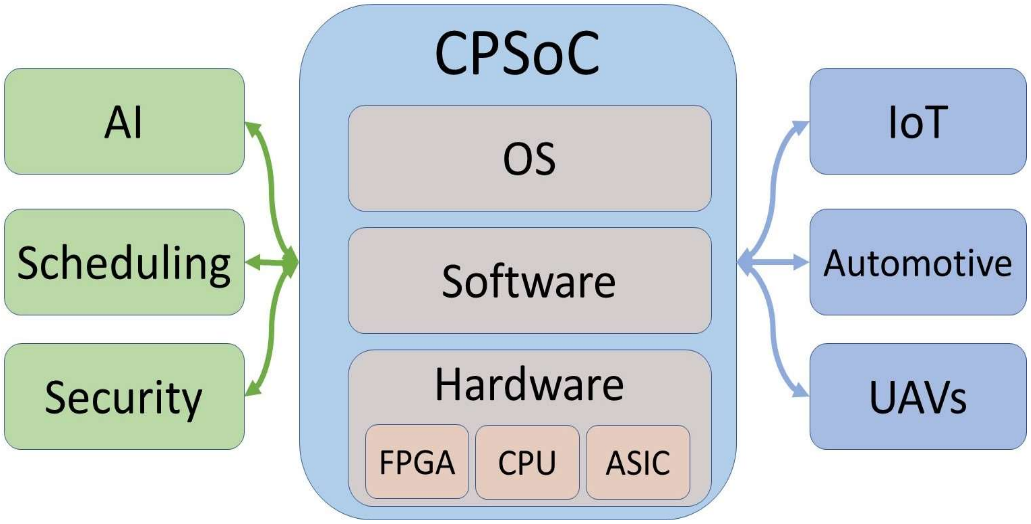
- Identifying new vulnerabilities of MPSoCs, which did not exist in traditional platforms
- Developing cost- and performance-effective methodologies to prevent or mitigate them
- Adopting security as a first-class citizen in
  - designing MPSoCs for MCS (secure-by design concept).
  - Scheduling techniques

Back to the Big Picture

Future Directions

Back to the Bigger Picture

Future Directions



CPSoC

OS

Software

Hardware

FPGA CPU ASIC

AI

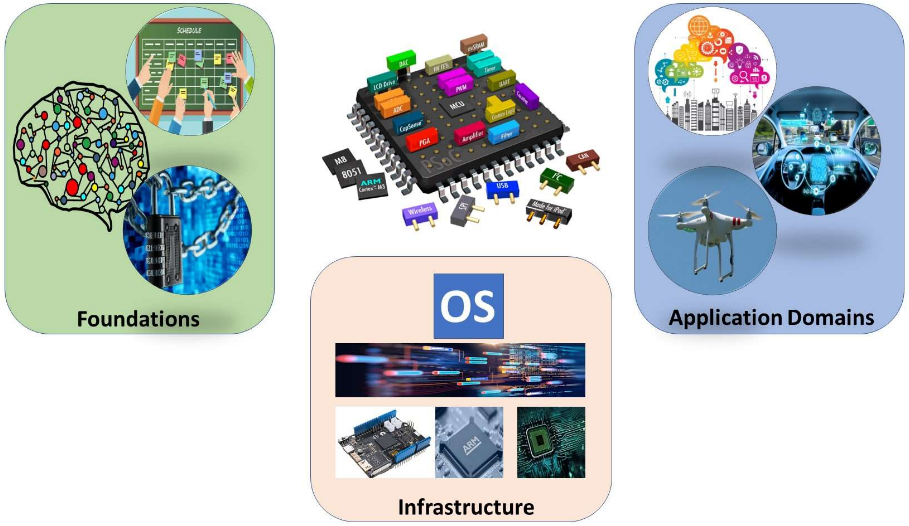
Scheduling

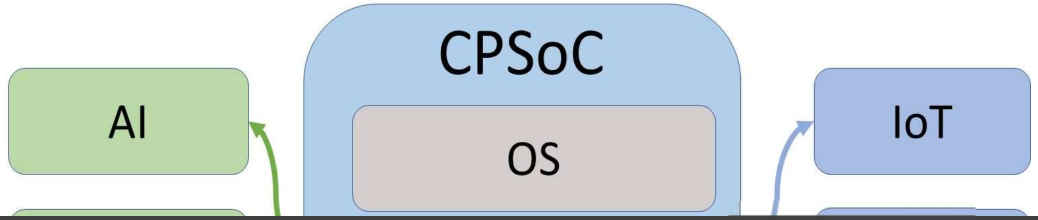
Security

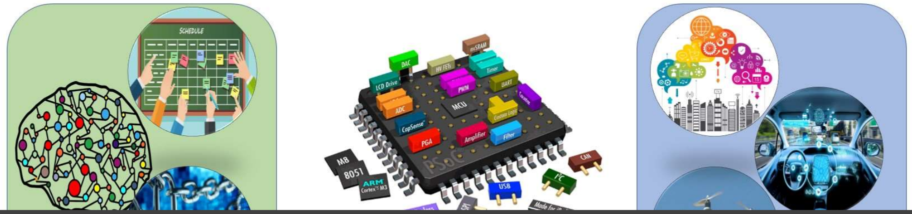
IoT

Automotive

UAVs







Intelligent Cyber-Physical Systems-on-Chip

Back to the Bigger Picture

Future Directions

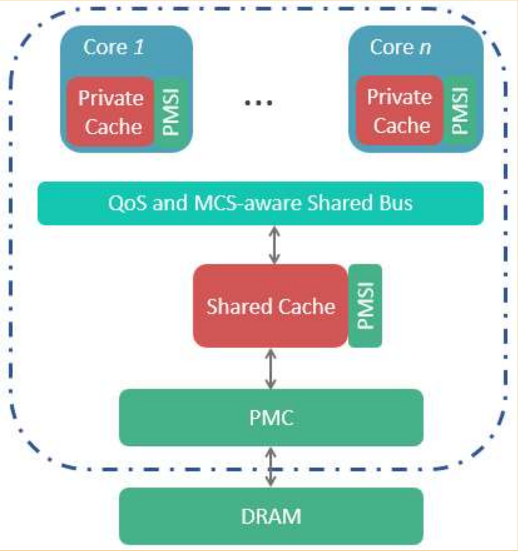
Predictable CPS

I propose architectures for predictable MPSoC-based CPS

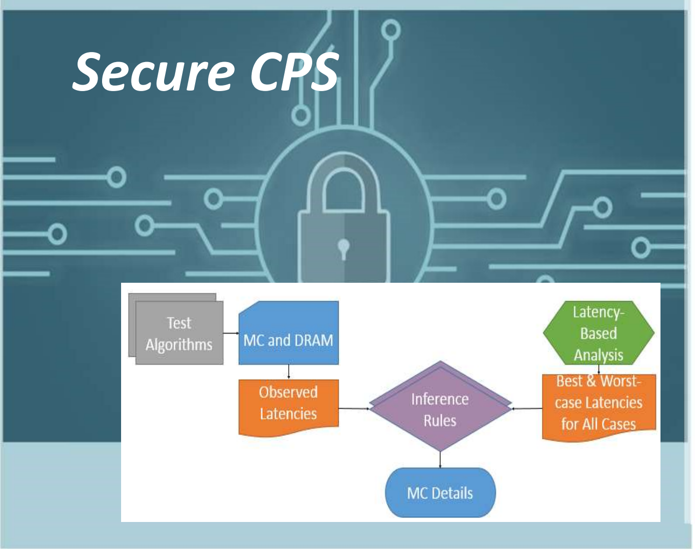
Predictable Shared Memory Hierarchy

PREMIUM

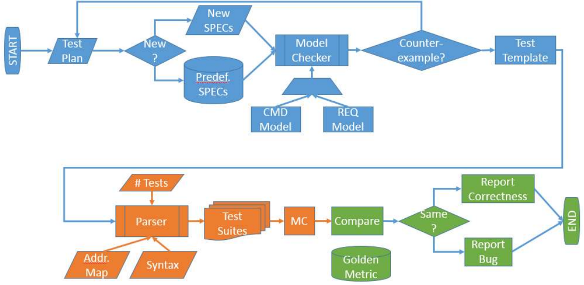
Predictable Memory Hierarchy



Secure CPS



Verified CPS


START
Test Plan
New ?
New SPECs
Predef. SPECs
Model Checker
CMD Model
REQ Model
Counter-example?
Test Template
# Tests
Parser
Addr. Map
Syntax
Test Suites
MC
Compare
Golden Metric
Same ?
Report Correctness
Report Bug
END