A High-Resilience Imprecise Computing Architecture for Mixed-Criticality Systems

Zhe Jiang[1][2], Xiaotian Dai[1], Alan Burns[1], Neil Audsley[3]
ZongHua Gu[4], Ian Gray[1]
[1]University of York, United Kingdom
[2]University of Cambridge, United Kingdom
[3]City, University of London, United Kingdom
[4]Umea University, Sweden

Outline

- Mixed-Criticality System (MCS)
- Motivation and Research Challenges: Imprecise MCS
- HIART-MCS: Method, Architecture, and Design
- Theoretical Model and Optimisation
- Evaluation
- Conclusion

Outline

- **Mixed-Criticality System (MCS)**
- Motivation and Research Challenges: Imprecise MCS
- HIART-MCS: Method, Architecture, and Design
- Theoretical Model and Optimisation
- Evaluation
- Conclusion

Safety Criticality Levels

- **Criticality level:** the required **safety assurance level** of system components.

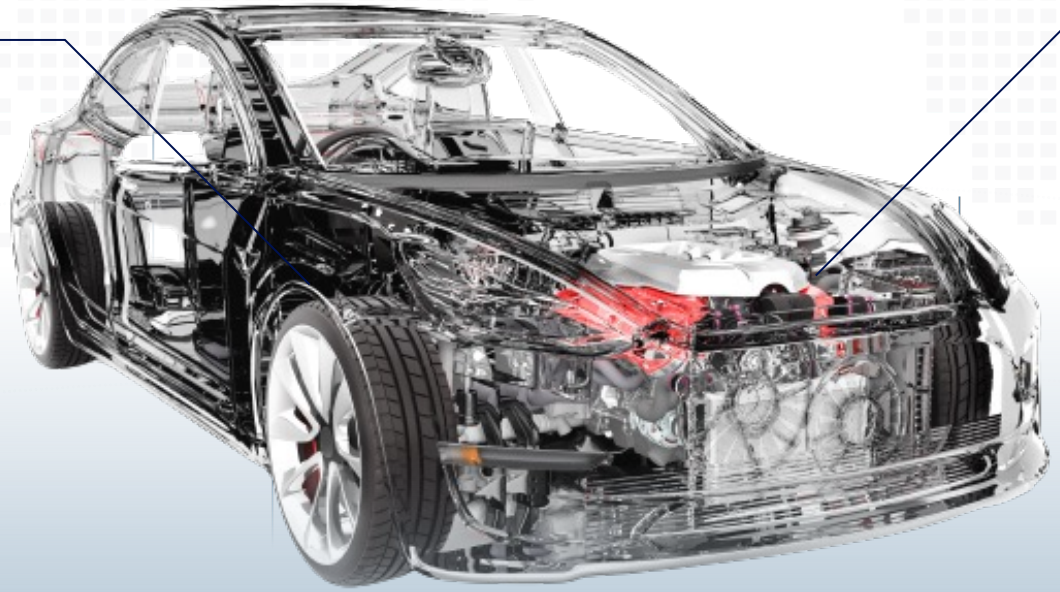
Safety Criticality Levels

- **Criticality level:** the required **safety assurance level** of system components.

Body Control System

Criticality Level:

Low



Engine Control System

Criticality Level:

High

Mixed-Criticality System (MCS)

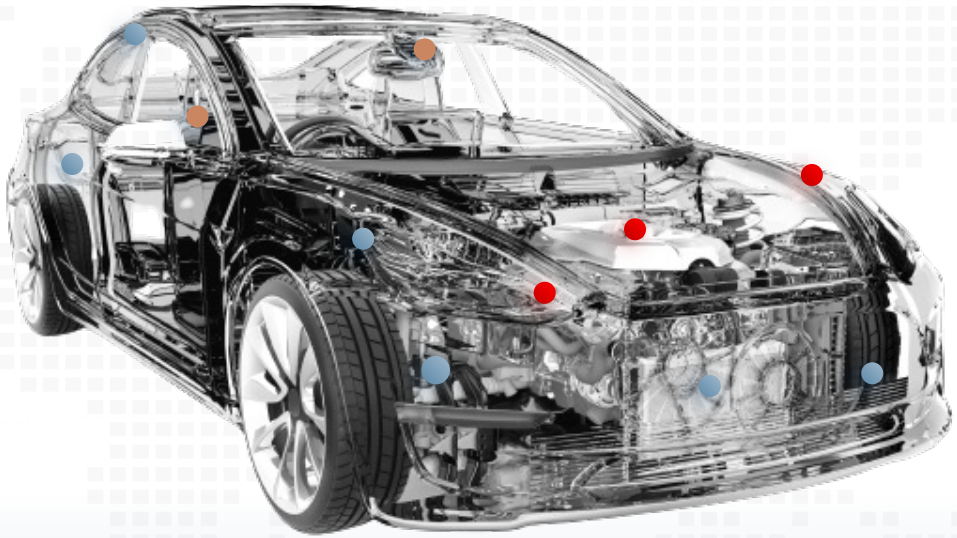
- Different critical modules are now implemented using different chips.

Suspension/Dumping
Level C

Body Control
Level B

Data Fusion
Level B, up to Level D

Electric Power Steering
Level C



SRR,MRR,LRR
Level D

Adaptive Cruise Control
Level D

Engine Management Unit
Level D

Battery Management(12V,48V,HV)
Level C

Transmission Control
Level D

DC-DC Converter
Level C

Motor (Alterno Starter,eAxel drive…)
Level C

Mixed-Criticality System (MCS)

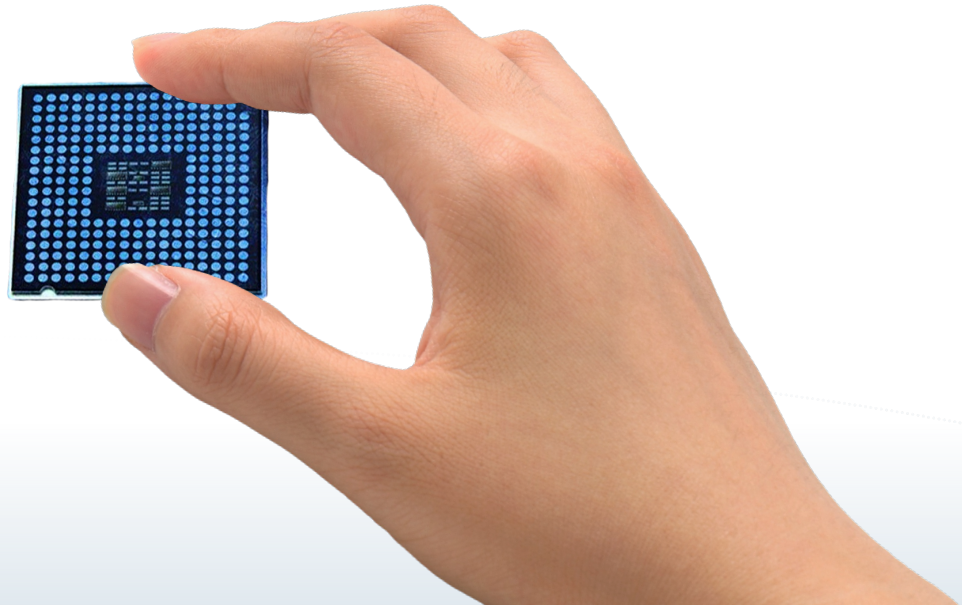
- Mixed-Criticality System: designing different critical components onto a **shared hardware** platform.

Key Benefits

- Overhead optimization
  - Size
  - Weight
  - Power
  - Cost
  - …
- System diversity & possibility

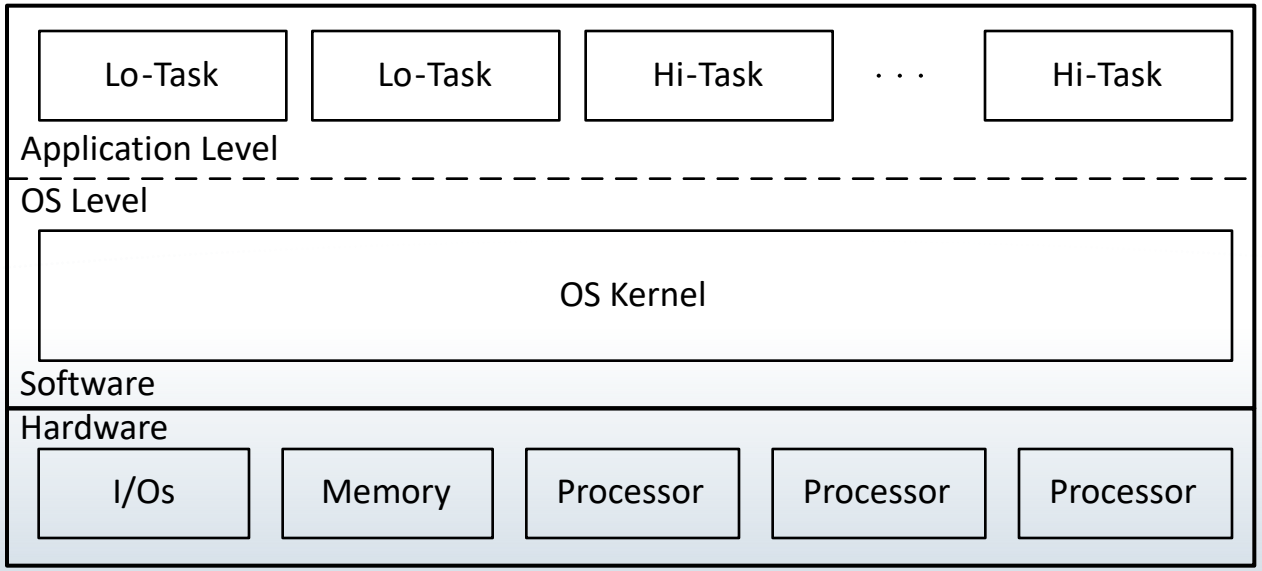
Final goal

- All systems on one chip!



General System Architecture of MCS

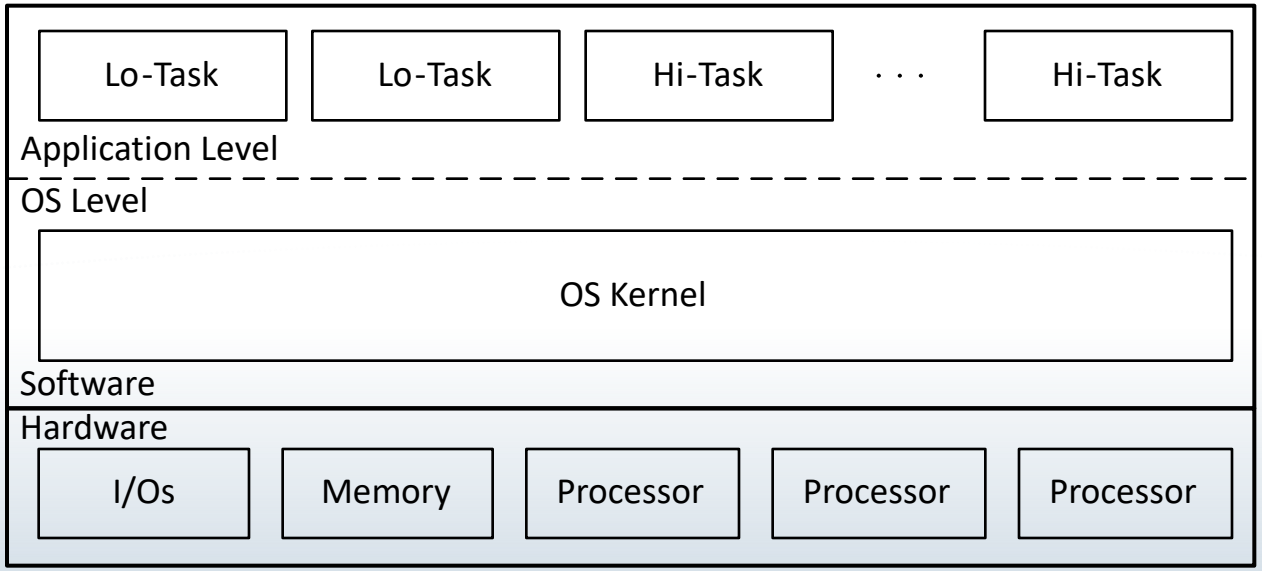
- Mixed-Criticality System: designing different critical components onto a **shared hardware** platform.
  - Lo-Task: Low-criticality task
  - Hi-Task: High-criticality task



A conventional MCS architecture

General System Architecture of MCS

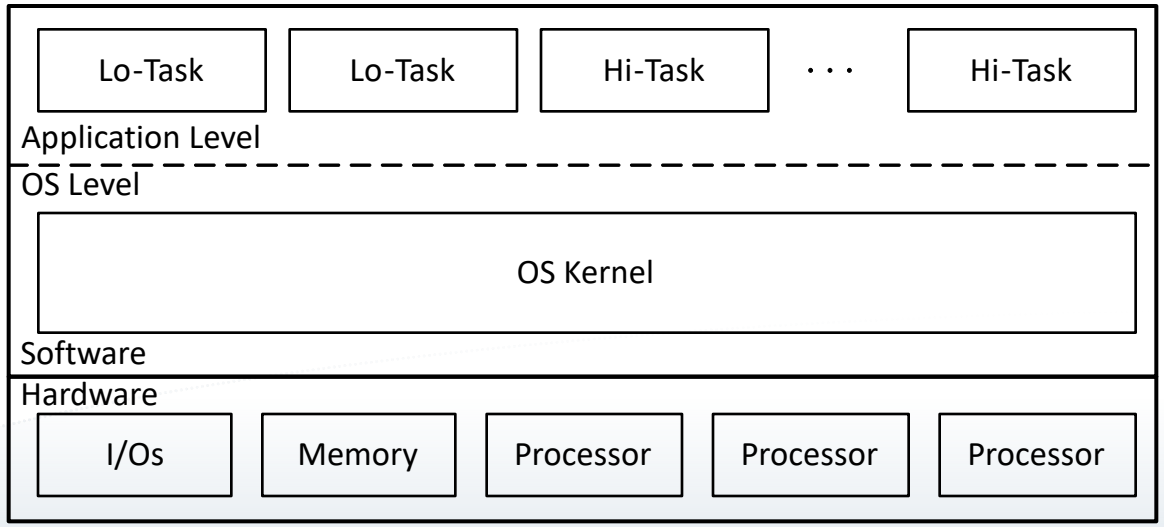
- Key problem
  - Resource contentions
  - Interferences
- In coping with these issues, the classic two-mode MCS was presented
  - S Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance [RTSS 2009]



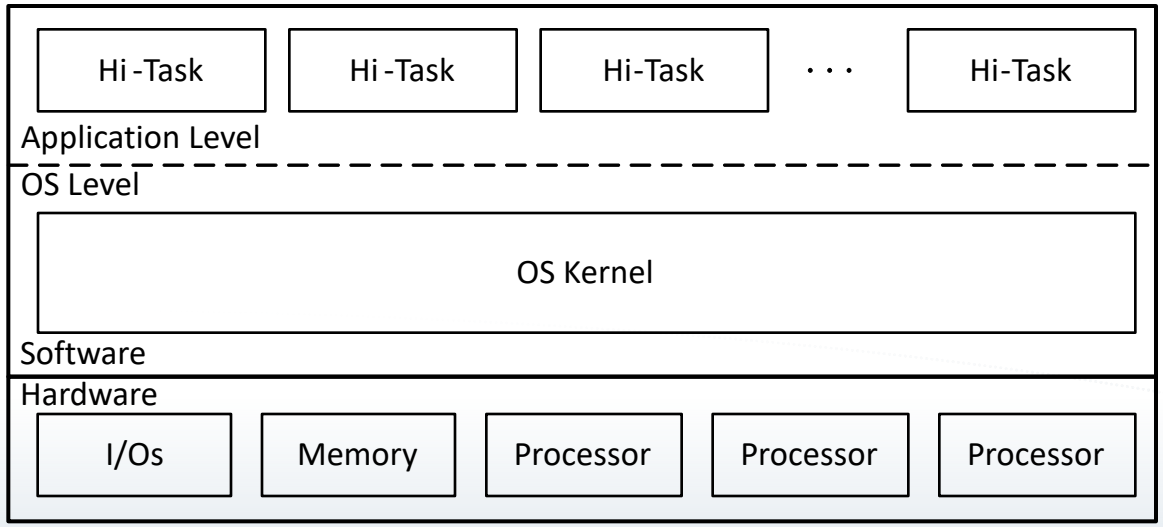
A conventional MCS architecture

Mode Switch in MCS

- Dual-mode MCS introduces two system modes:
  - Lo-mode: all tasks are executed
  - Hi-mode: only Hi-tasks are executed and Lo-tasks are terminated



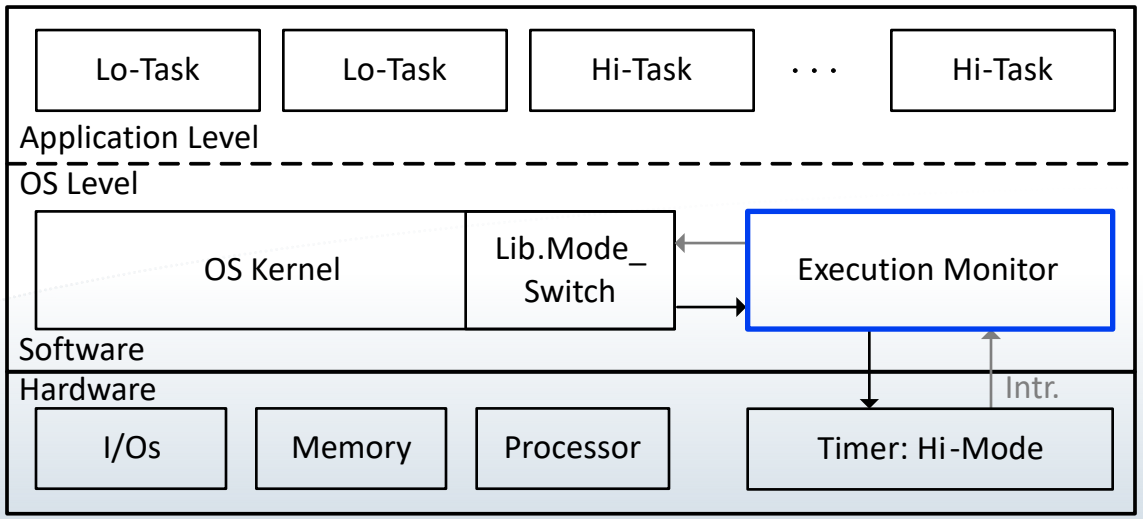
System in Lo-Mode



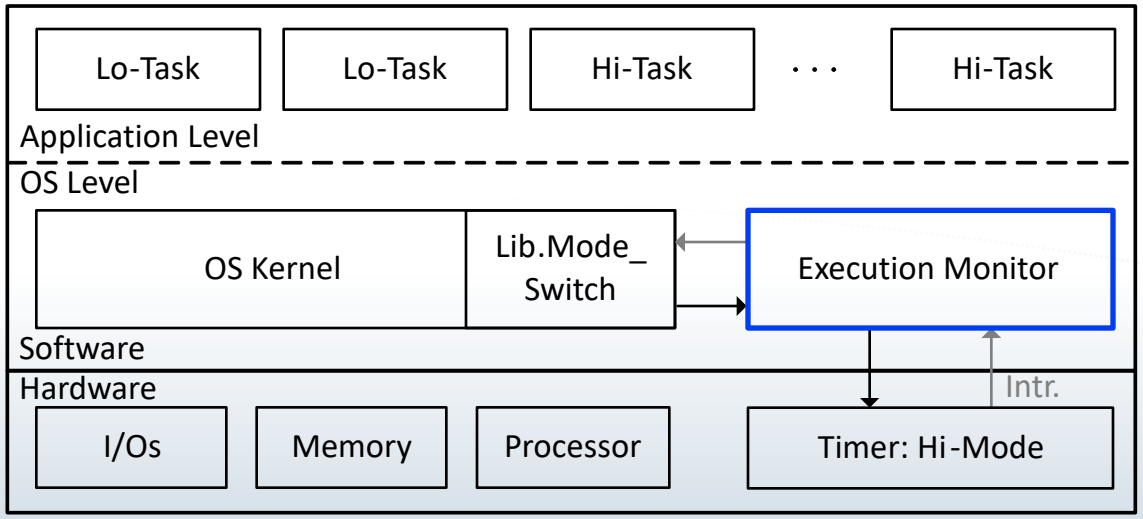
System in Hi-Mode

Mode Switch in MCS

- To achieve this, a system monitor is required
  - In OS kernel, or
  - As an independent hypervisor
- Many practical system frameworks were built upon this model
  - West et al. A Virtualized Separation Kernel for Mixed-Criticality Systems [TOCS, 2016]
  - Kim et al. Supporting I/O and IPC via fine-grained OS isolation for mixed-criticality tasks [RTNS 2018]
  - Gadepalli et al. Chaos: a System for Criticality-Aware, Multi-core Coordination [RTAS, 2019]



Lo-Mode



Hi-Mode

Outline

- Mixed-Criticality System (MCS)
- **Motivation and Research Challenges: Imprecise MCS**
- HIART-MCS: Method, Architecture, and Design
- Theoretical Model and Optimisation
- Evaluation
- Conclusion

Potential Safety Hazards in MCS and Imprecise MCS

- As reported, deploying safety mode switches may cause safety risks
  - A. Burns and S.Baruah. Towards a more practical model for mixed criticality systems [WMC, 2013]
  - S.Baruah et al. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors [ECRTS 2016]

Potential Safety Hazards in MCS and Imprecise MCS

- As reported, deploying safety mode switches may cause safety hazards
  - A. Burns and S.Baruah. Towards a more practical model for mixed criticality systems [WMC, 2013]
  - S.Baruah et al. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors [ECRTS 2016]

- **Approximation** is an effective mitigation of this problem (i.e., Imprecise MCS, IMCS):
  - Executing Lo-tasks with decreased computing precision and less time budget
  - Existing theoretical models for IMCS
    - L. Huang. Graceful degradation of low-criticality tasks in multiprocessor dual-criticality systems [RTNS 2018]
    - D.Liu. Scheduling analysis of imprecise mixed-criticality real-time tasks [TC 2018]
    - R.M.Pathan. Improving the quality-of-service for scheduling MCSs on multiprocessor [ECRTS 2017]
    - X. Gu. Dynamic budget management and budget reclamation for mixed-criticality systems [RTS 2019]

- However, a systematic system framework of IMCS is still missing…

Research Challenges of Building a practical IMCS

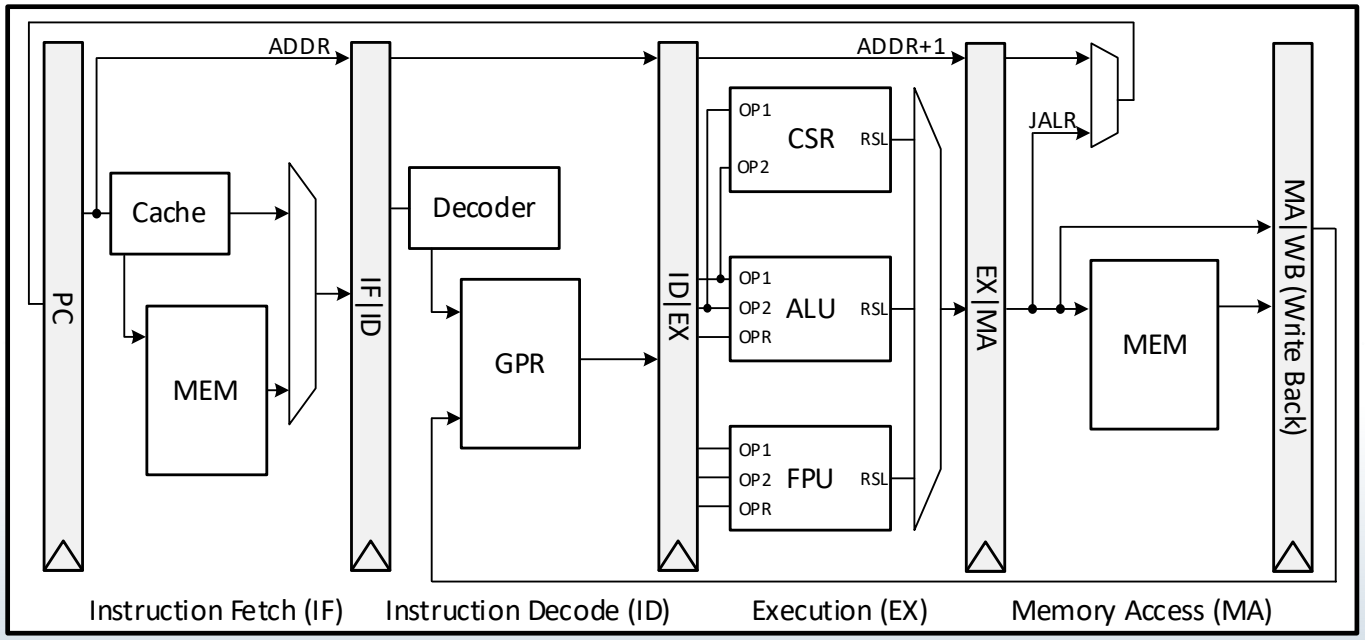
- Research Challenges (**RC.x**) whiling building a practical IMCS
  - **RC.1.** an effective method to achieve approximation of the Lo-tasks
  - **RC.2.** a timely method to configure the approximation degree at run-time
  - **RC.3.** a quantitative analysis to determine the appropriate approximation degree for each lo-task
  - **RC.4.** a systematic solution to realise the new features introduced by IMCS

Outline

- Mixed-Criticality System (MCS)
- Motivation and Research Challenges: Imprecise MCS
- **HIART-MCS: Method, Architecture, and Design**
- Theoretical Model and Optimisation
- Evaluation
- Conclusion

RC.1. Achieving approximation

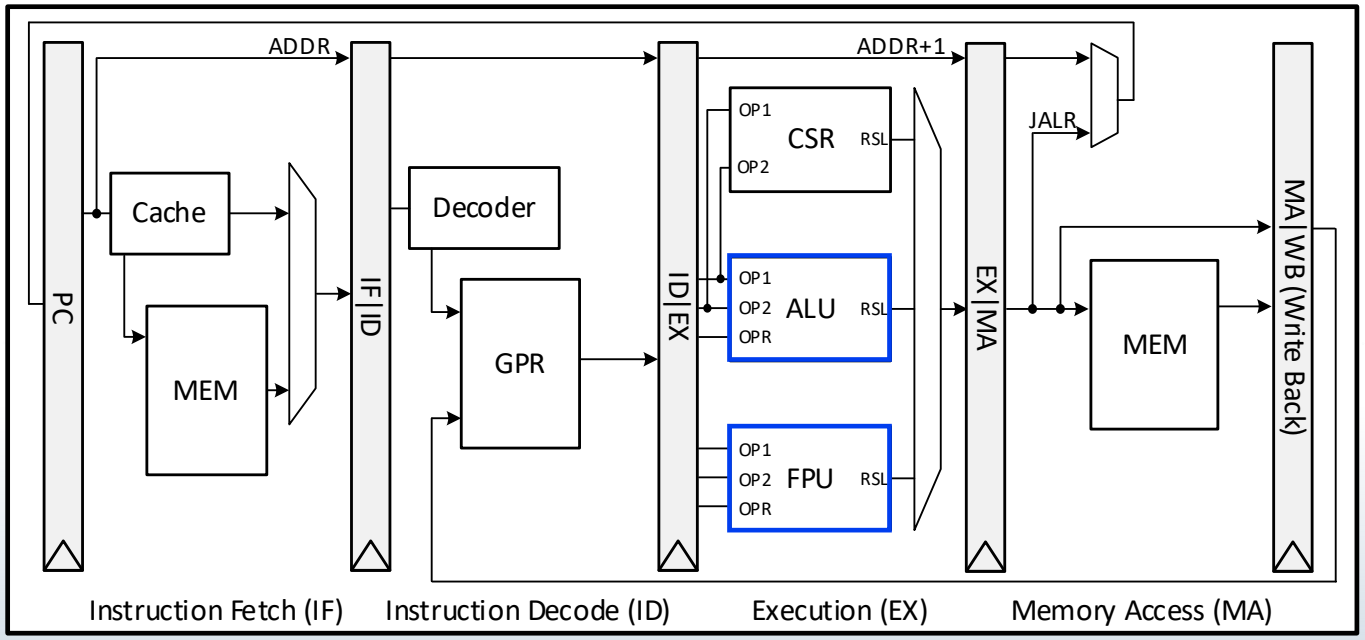
- The working procedures of a processor:
  - Instruction Fetch (IF)
  - Instruction Decode (ID)
  - Execution (EX)
  - Memory Access (MA)
  - Write-Back (WB)



Top-level micro-architecture of a 5-stage pipelined RISC-V processor

RC.1. Achieving approximation

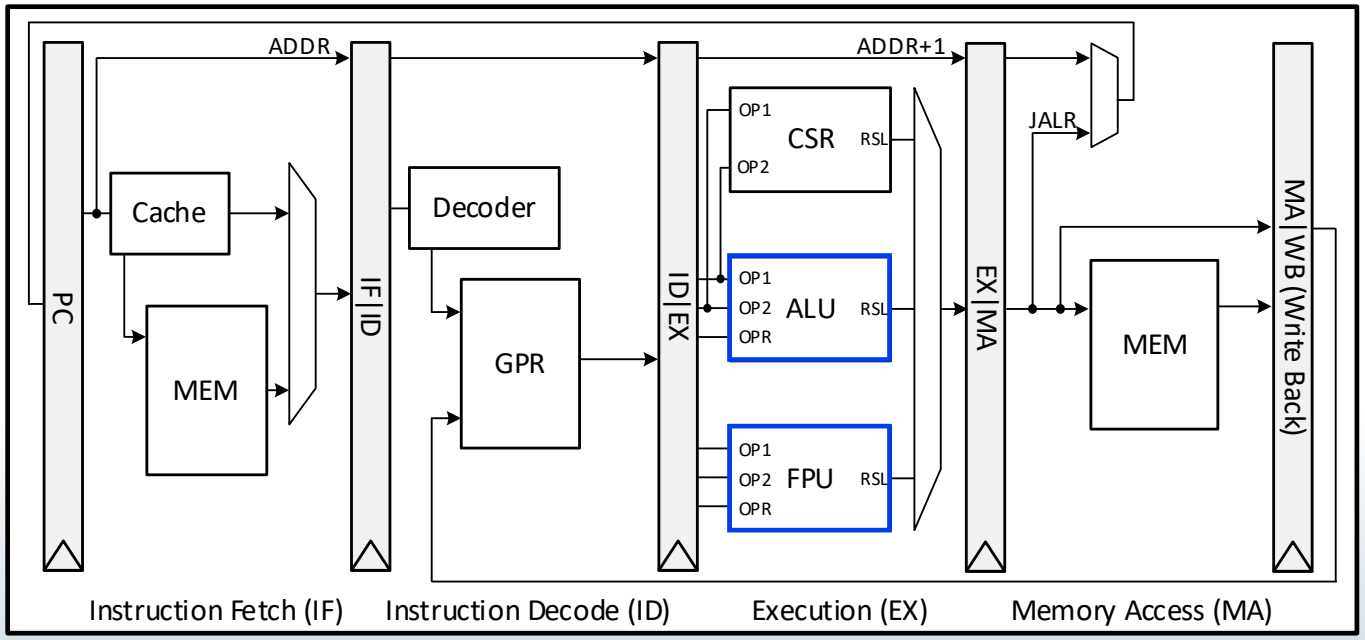
- The working procedures of a processor:
  - Instruction Fetch (IF)
  - Instruction Decode (ID)
  - Execution (EX)
  - Memory Access (MA)
  - Write-Back (WB)



Top-level micro-architecture of a 5-stage pipelined RISC-V processor

RC.1. Achieving approximation

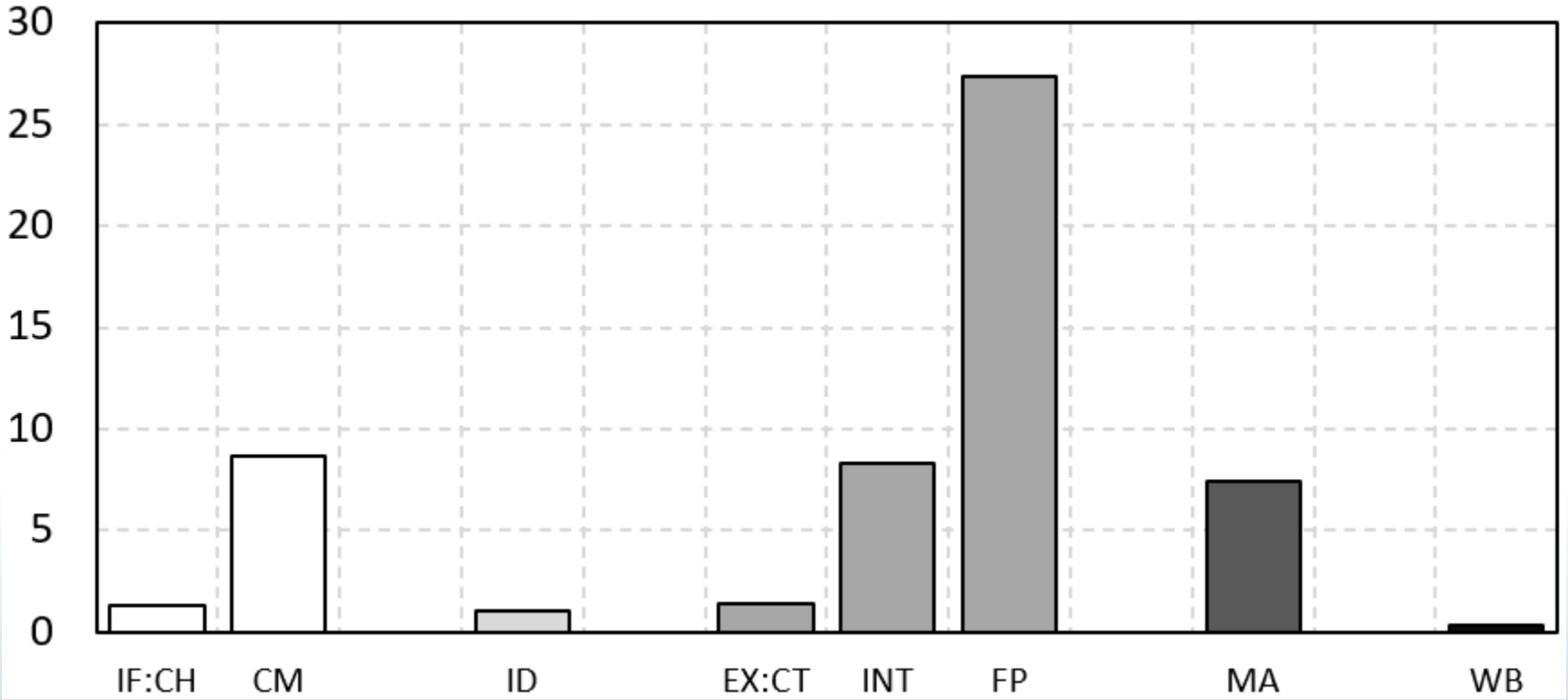
- To verify our assumption,
  - Randomly generated and executed 1,000 instructions
  - Executed the generated instructions and record the execution time



Top-level micro-architecture of a 5-stage pipelined RISC-V processor

RC.1. Achieving approximation

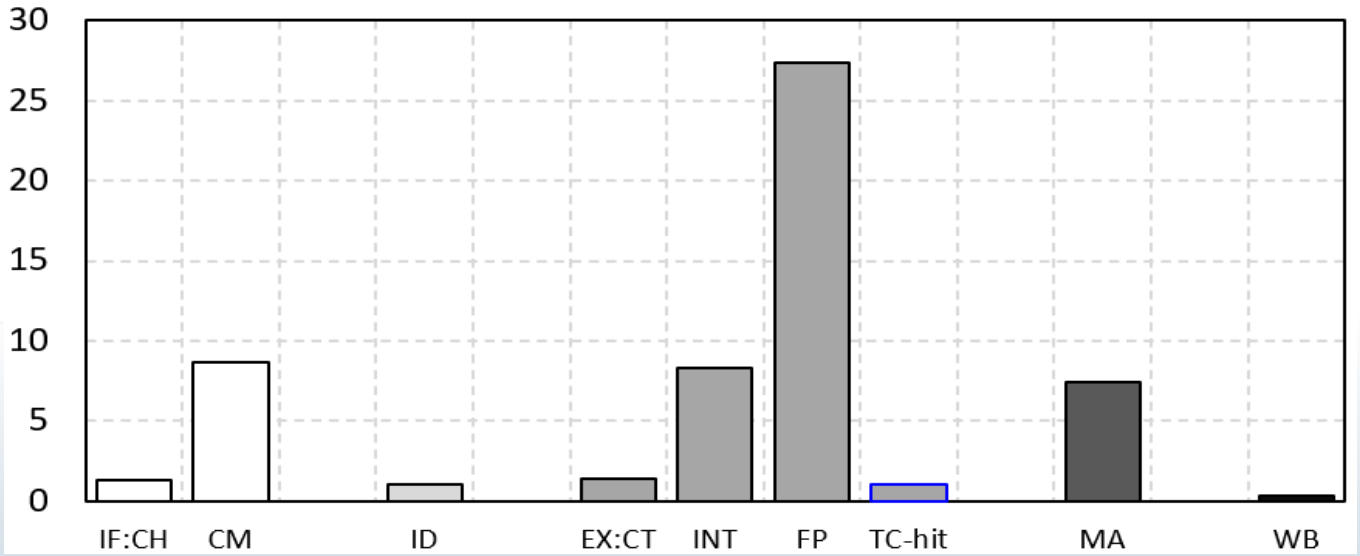
- To verify our assumption,
  - Randomly generated and executed 1,000 instructions
  - Executed the generated instructions and record the execution time
- The floating point (FP) computation dominates an instruction's execution time



Experimental Results

RC.1. Achieving approximation

- FP computation is returned in a single clock cycle when it meets a trivial case (TC)
  - E.g., 0 x Y = 0



Experimental Results

Index #	OPR	OP1	OP2	RD
1	ADD (+)	0	X	X
2	ADD (+)	X	0	X
3	ADD (+)	X	-X	0
4	SUB (-)	X	0	X
5	SUB (-)	0	X	-X
6	SUB (-)	X	X	0
7	MULT (×)	X	0	0
8	MULT (×)	0	X	0
9	MULT (×)	X	±1	±X
10	MULT (×)	±1	X	±X
11	DIV (÷)	0	X	0
12	DIV (÷)	X	±1	±X

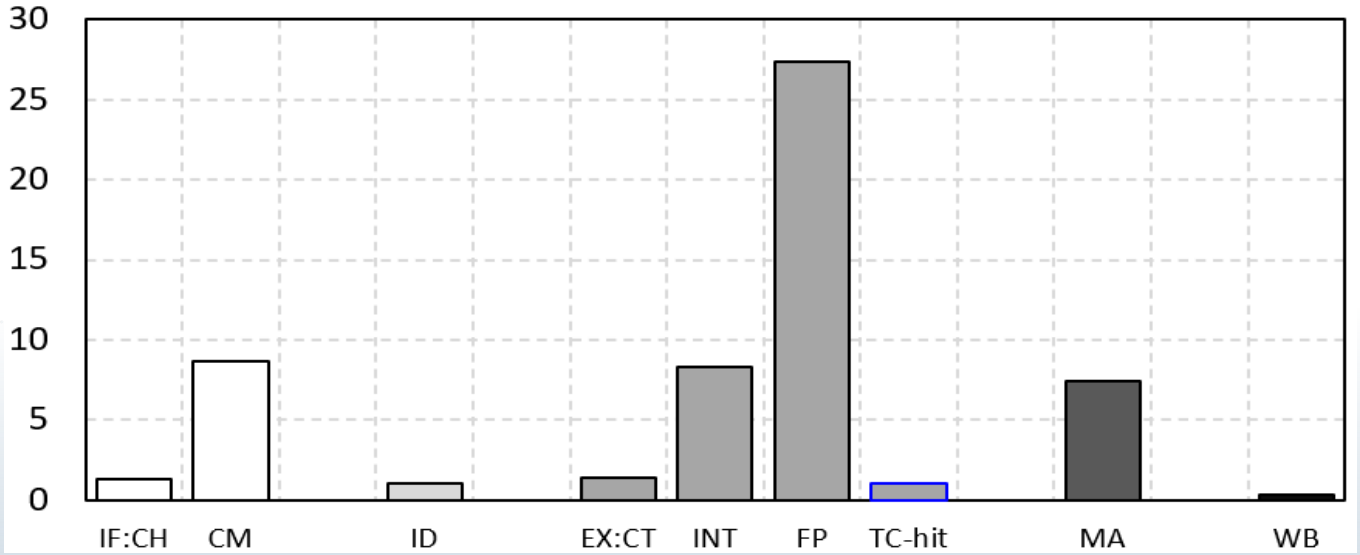
Travail Case (TC)

RC.1. Achieving approximation

- FP computation is returned in a single clock cycle when it meets a trivial case (TC)
  - E.g., 0 x Y = 0
- Achieving approximation at the FPU
  - Cutting down the valid bit-width of input operands, e.g.,

$$1.071 \times 1.001 \ \textbf{(TC-miss)} \approx 1.07 \times 1.00 \ \textbf{(TC-hit)} = 1.07$$

- While designing the new processor, RC.2 is recalled:
  - Supporting run-time configurations of the approximation degree



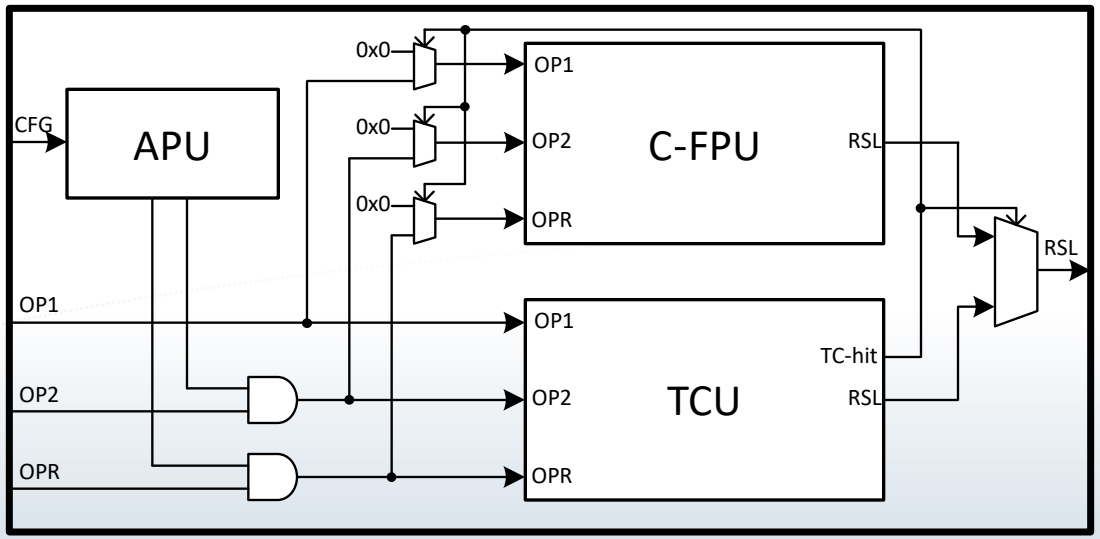
Experimental Results

Index #	OPR	OP1	OP2	RD
1	ADD (+)	0	X	X
2	ADD (+)	X	0	X
3	ADD (+)	X	-X	0
4	SUB (-)	X	0	X
5	SUB (-)	0	X	-X
6	SUB (-)	X	X	0
7	MULT (×)	X	0	0
8	MULT (×)	0	X	0
9	MULT (×)	X	±1	±X
10	MULT (×)	±1	X	±X
11	DIV (÷)	0	X	0
12	DIV (÷)	X	±1	±X

Travail Case (TC)

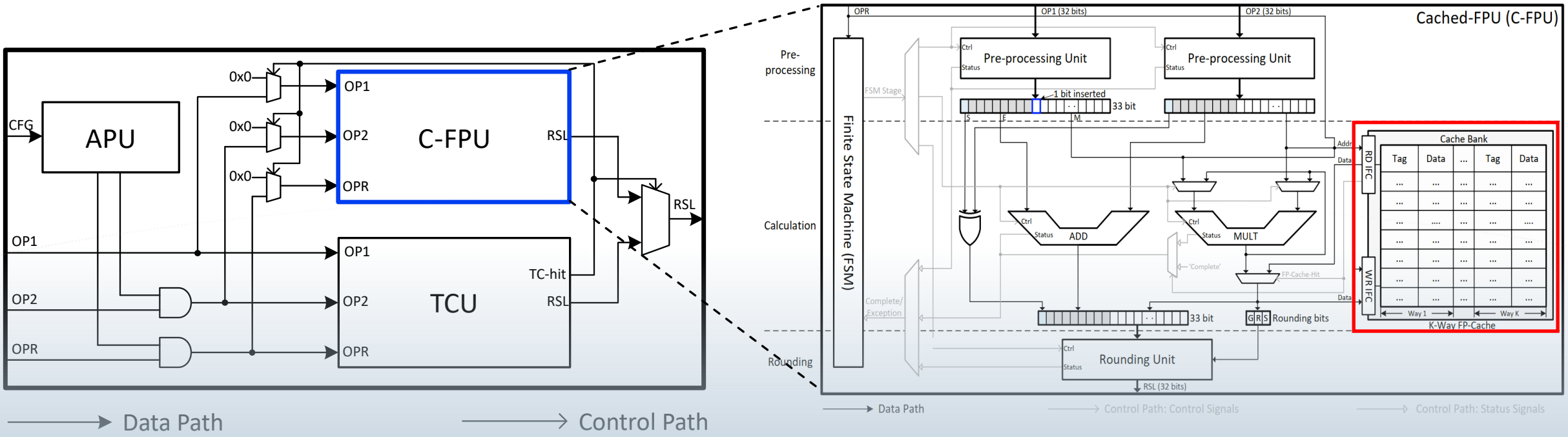
HIART-FPU

- The top-level design of the proposed FPU contains three modules:
  - Cached-FPU (C-FPU) – executes FP calculations if TC-miss
  - Trivial Computation Unit (TCU) – executes FP calculation if TC-hit
  - Approximation Unit (APU) – controls the approximation degree of input data



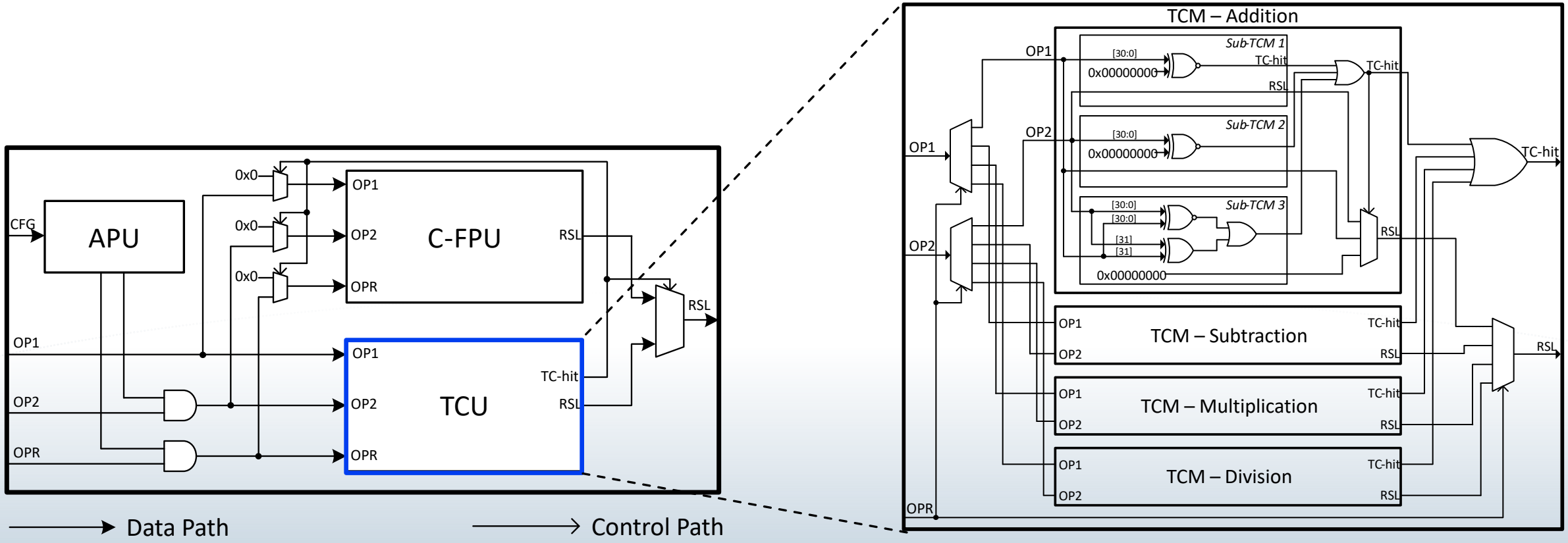
HIART-FPU

- The top-level design of the proposed FPU contains three modules
  - Cached-FPU (C-FPU) – executes FP calculations if TC-miss
  - Trivial Computation Unit (TCU) – executes FP calculation if TC-hit
  - Approximation Unit (APU) – controls the approximation degree of input data



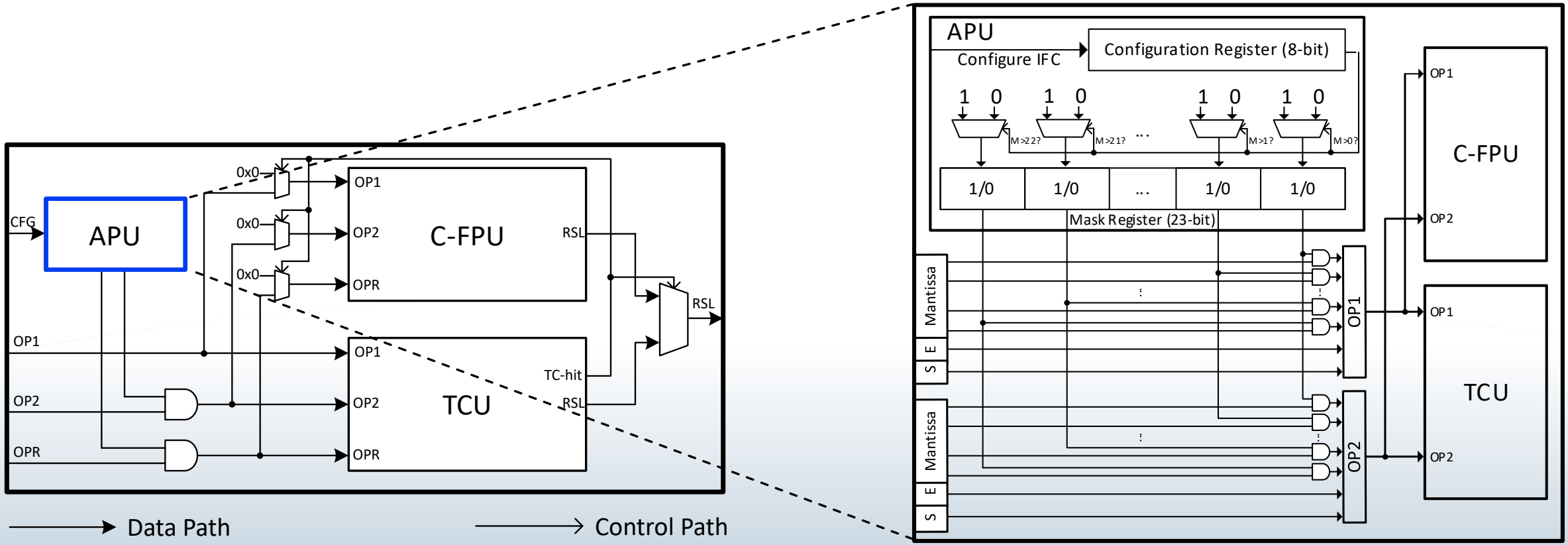
HIART-FPU

- The top-level design of the proposed FPU contains three modules
  - Cached-FPU (C-FPU) – executes FP calculations if TC-miss
  - **Trivial Computation Unit (TCU) – executes FP calculation if TC-hit**
  - Approximation Unit (APU) – controls the approximation degree of input data



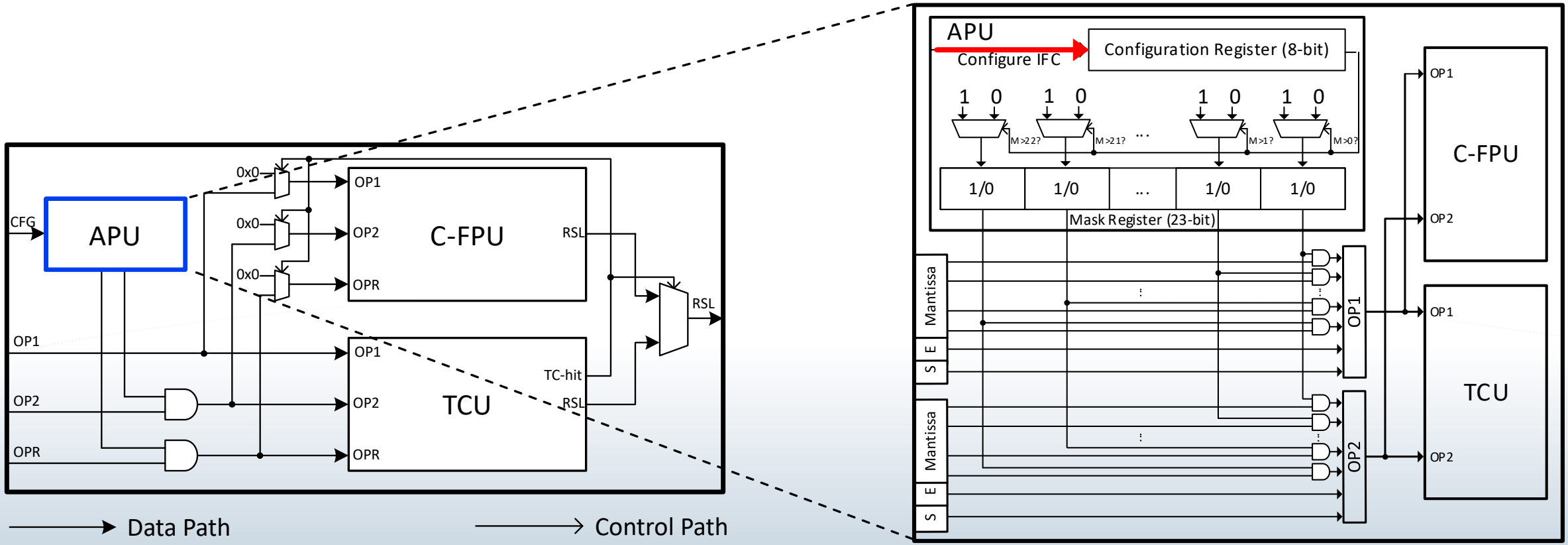
HIART-FPU

- The top-level design of the proposed FPU contains three modules
  - Cached-FPU (C-FPU) – executes FP calculations if TC-miss
  - Trivial Computation Unit (TCU) – executes FP calculation if TC-hit
  - **Approximation Unit (APU) – controls the approximation degree of input data**



HIART-FPU

- The top-level design of the proposed FPU contains three modules
  - Cached-FPU (C-FPU) – executes FP calculations if TC-miss
  - Trivial Computation Unit (TCU) – executes FP calculation if TC-hit
  - **Approximation Unit (APU) – controls the approximation degree of input data**



Research Challenges of Building a practical IMCS

- Research Challenges (**RC.x**) whiling building a practical IMCS
  - **RC.1.** an effective method to achieve approximation of the Lo-tasks
  - **RC.2.** a timely method is needed to configure the approximation degree at run-time
  - **RC.3.** a quantitative analysis to determine the appropriate approximation degree for each Lo-task
  - **RC.4.** a systematic solution to realise the new features introduced by IMCS

Measurement-based method

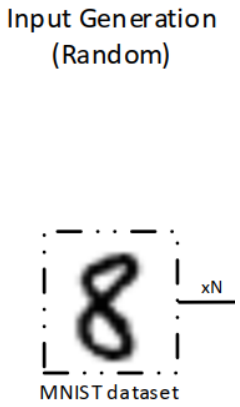
- A measurement-based method is aiming to find Lo-task (τ_i)
  - Approximation degree (M_i) of the Lo-task
  - WCET with approximation (C_i^{AP})

Measurement-based method

- A measurement-based method is aiming to find Lo-task (τ_i)
  - Approximation degree (M_i) of the Lo-task
  - WCET with approximation (C_i^{AP})
- The measurement-based method contains three steps:
  - Step 1: Initialisation and input generation
  - Step 2: Experimental measures
  - Step 3: Results analysis

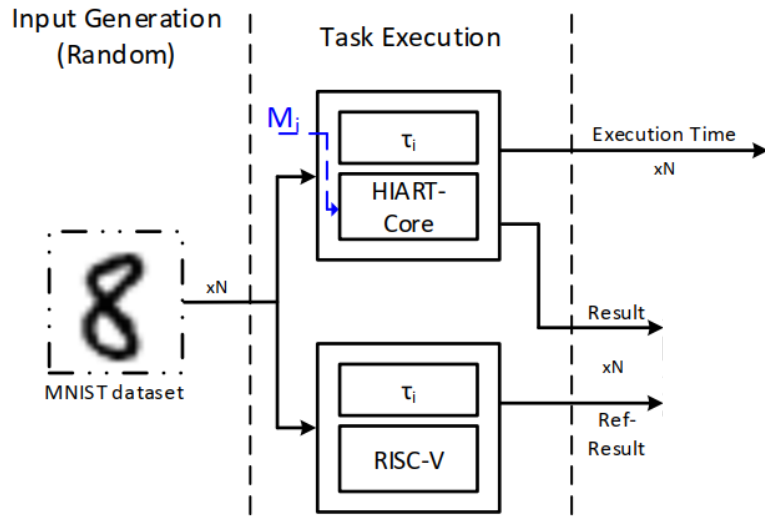
Measurement-based method

- A measurement-based method is aiming to find Lo-task (τ_i)
  - Approximation degree (M_i) of the Lo-task
  - WCET with approximation (C_i^{AP})
- The measurement-based method contains three steps:
  - **Step 1: Initialisation and input generation**
  - Step 2: Experimental measures
  - Step 3: Results analysis



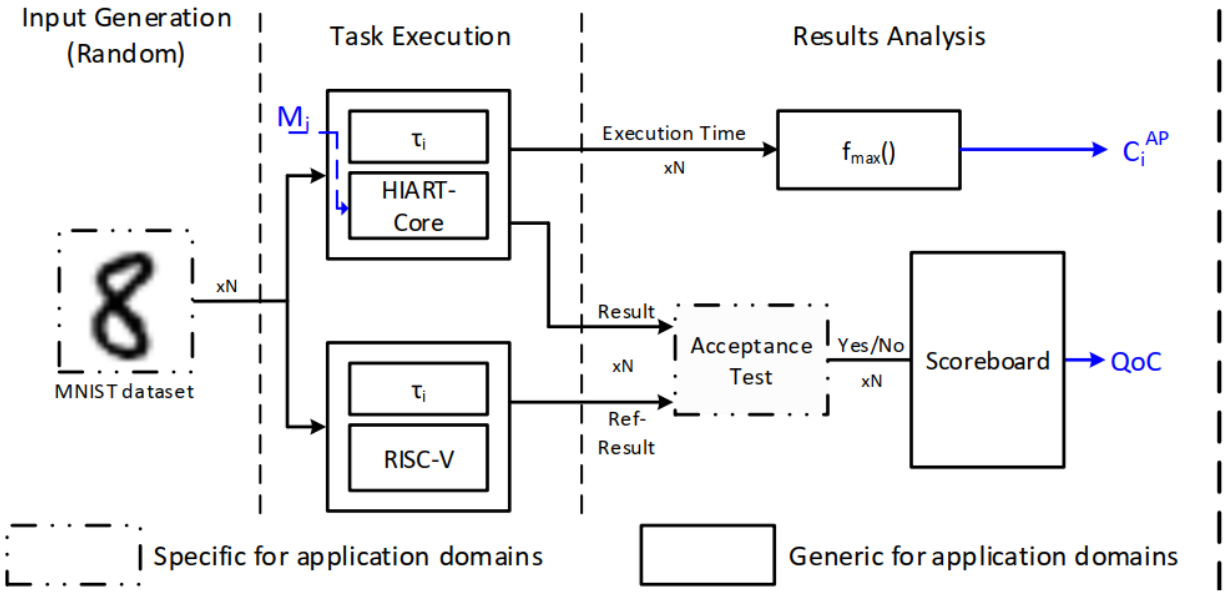
Measurement-based method

- A measurement-based method is aiming to find Lo-task (τ_i)
  - Approximation degree (M_i) of the Lo-task
  - WCET with approximation (C_i^{AP})
- The measurement-based method contains three steps:
  - Step 1: Initialisation and input generation
  - **Step 2: Experimental measures**
  - Step 3: Results analysis



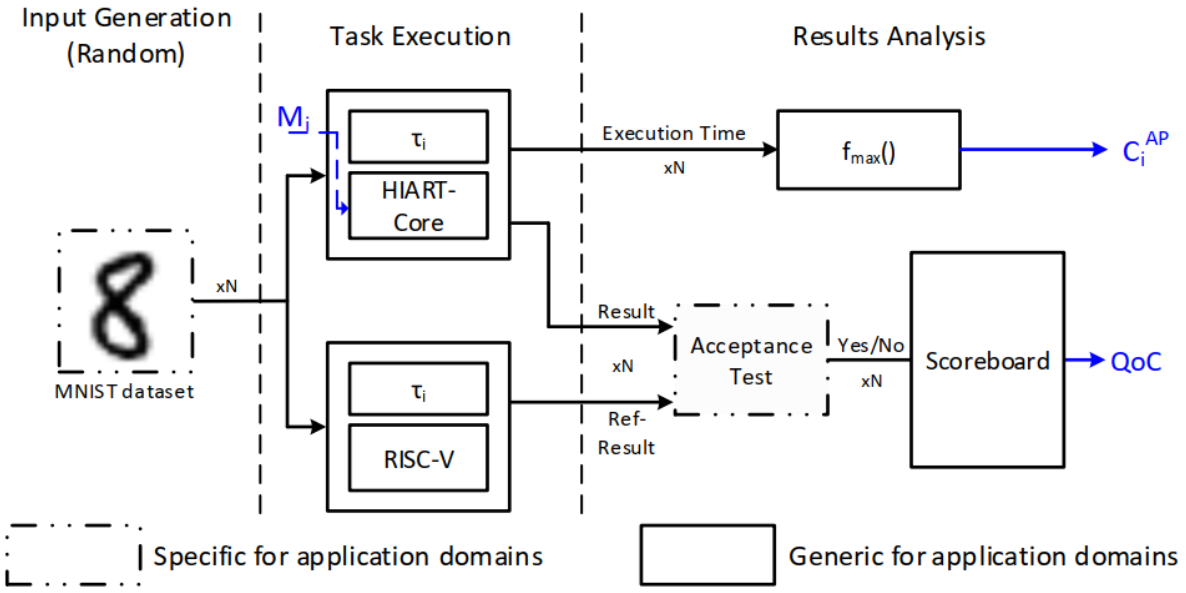
Measurement-based method

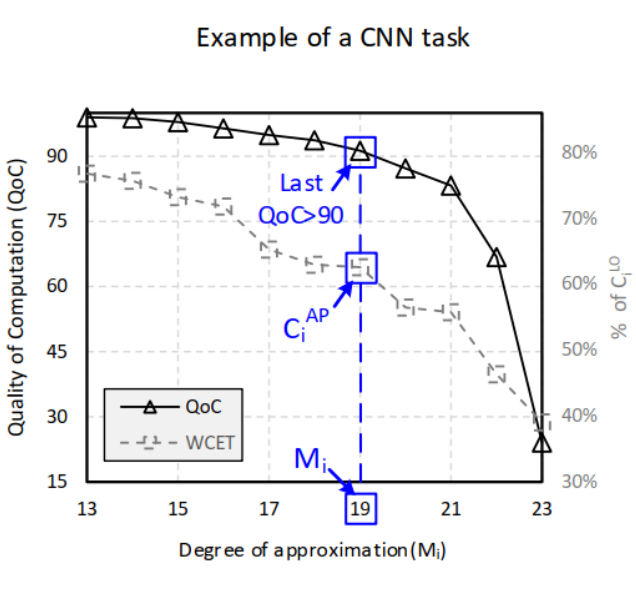
- A measurement-based method is aiming to find Lo-task (τ_i)
  - Approximation degree (M_i) of the Lo-task
  - WCET with approximation (C_i^{AP})
- The measurement-based method contains three steps:
  - Step 1: Initialisation and input generation
  - Step 2: Experimental measures
  - **Step 3: Results analysis**



Measurement-based method

- A measurement-based method is aiming to find Lo-task (τ_i)
  - Approximation degree (M_i) of the Lo-task
  - WCET with approximation (C_i^{AP})
- The measurement-based method contains three steps:
  - Step 1: Initialisation and input generation
  - Step 2: Experimental measures
  - **Step 3: Results analysis**



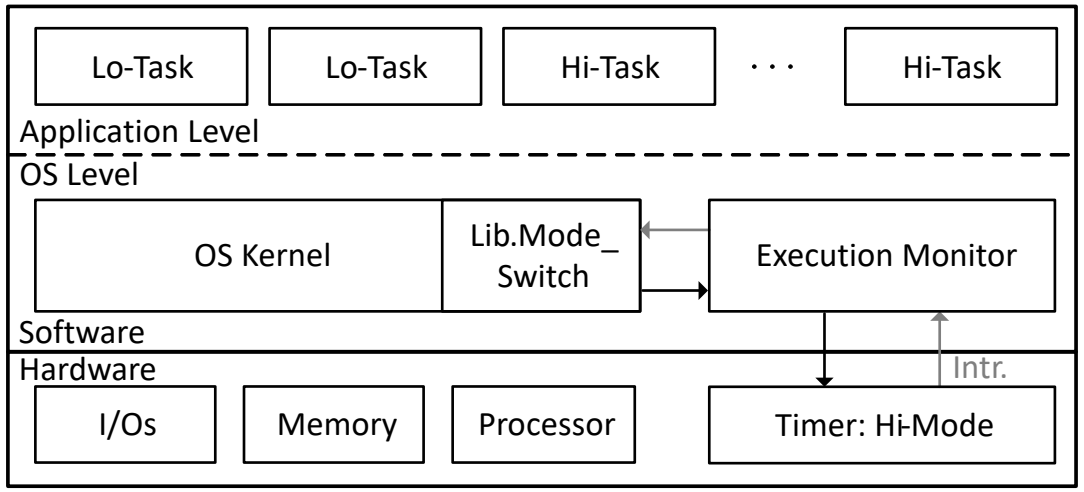


Research Challenges of Building a practical IMCS

- Research Challenges (**RC.x**) whiling building a practical IMCS
  - **RC.1.** an effective method to achieve approximation of the Lo-tasks
  - **RC.2.** a timely method is needed to configure the approximation degree at run-time
  - **RC.3.** a quantitative analysis to determine the appropriate approximation degree for each lo-task
  - **RC.4.** a systematic solution to realize the new features introduced by IMCS

HIART-MCS: System Architecture

- Conventional MCS architecture
  - Two system modes
    - Lo-Mode: Lo-tasks + Hi-tasks
    - Hi-Mode: Hi-tasks
  - Software tasks are managed by an OS
  - Executions are monitored by a monitor

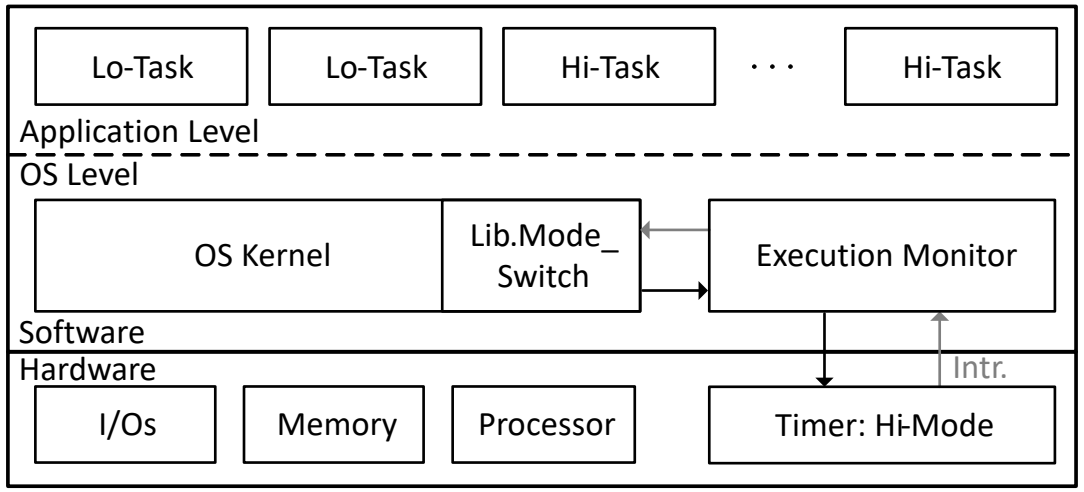


System Architecture of Conventional MCS

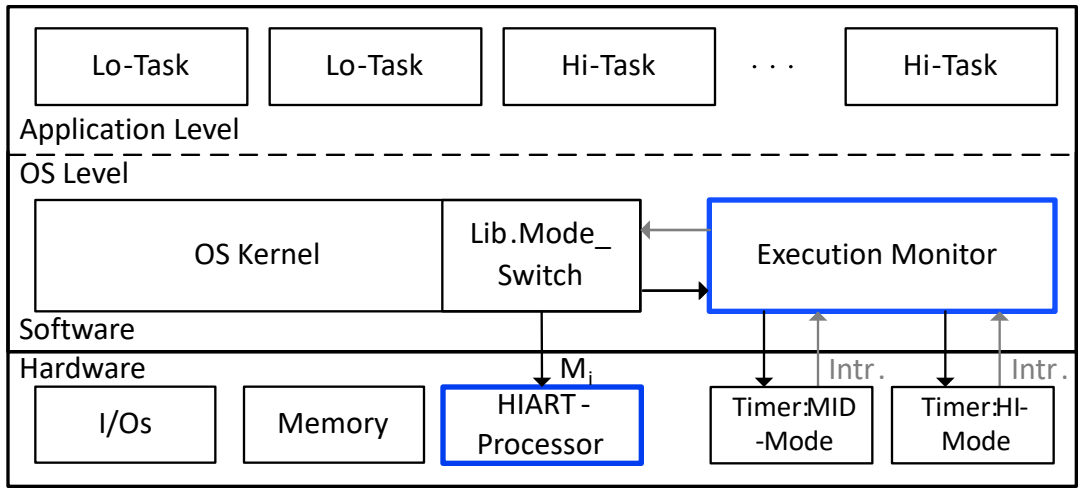
HIART-MCS: System Architecture

- Conventional MCS architecture
  - Two system modes
    - Lo-Mode: Lo-tasks + Hi-tasks
    - Hi-Mode: Hi-tasks
  - Software tasks are managed by an OS
  - Executions are monitored by a monitor

- HIHART MCS architecture
  - Three system modes
    - Lo-Mode: Lo-tasks + Hi-tasks
    - **Mid-Mode: Lo-tasks (AP) + Hi-tasks**
    - Hi-Mode: Hi-tasks
  - Software: modifying the execution monitor
  - Hardware: deploying HIART-processor
  - Source Compatibility



System Architecture of Conventional MCS



System Architecture of HIART-MCS

Outline

- Mixed-Criticality System (MCS)
- Motivation and Research Challenges: Imprecise MCS
- HIART-MCS: Method, Architecture, and Design
- **Theoretical Model and Optimisation**
- Evaluation
- Conclusion

Theoretical Model and Optimisation

- To support the newly introduced Mid-Mode, we need
  - (a) To have a corresponding timing analysis;
  - (b) A way to work out the best switching instances.
- For (a), we follow the analysis AMC-rtb, and adapted a response time analysis for our triple-mode system:

$$R_i = C_i^{\mathrm{MI}} + \sum_{j \in hpH(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^{\mathrm{MI}} + \sum_{j \in hpL(i)} \left\lceil \frac{R_i^{\mathrm{LO}}}{T_j} \right\rceil C_j^{\mathrm{LO}} + \sum_{j \in hpL(i)} \left\lceil \frac{R_i - R_i^{\mathrm{LO}}}{T_j} \right\rceil C_j^{\mathrm{AP}}$$

Eq 1. Response time of Hi-tasks from Lo- to Mid- mode

$$R_i = C_i^{\mathrm{HI}} + \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^{\mathrm{HI}} + \sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i^{MI}}{T_k} \right\rceil C_k^{\mathrm{AP}}$$

Eq 2. Response time of Hi-tasks from Mid- to Hi- mode (Lo-tasks are terminated)

Theoretical Model and Optimisation

- To support the newly introduced Mid-Mode, we need
  - (a) To have a corresponding timing analysis;
  - (b) A way to work out the best switching instances.
- For (a), we follow the analysis AMC-rtb, and adapted a response time analysis for our triple-mode system:

$$R_i = C_i^{\mathrm{MI}} + \sum_{j \in hpH(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^{\mathrm{MI}} + \sum_{j \in hpL(i)} \left\lceil \frac{R_i^{\mathrm{LO}}}{T_j} \right\rceil C_j^{\mathrm{LO}} + \sum_{j \in hpL(i)} \left\lceil \frac{R_i - R_i^{\mathrm{LO}}}{T_j} \right\rceil C_j^{\mathrm{AP}}$$

Eq 1. Response time of Hi-tasks from Lo- to Mid- mode

$$R_i = C_i^{\mathrm{HI}} + \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^{\mathrm{HI}} + \sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i^{MI}}{T_k} \right\rceil C_k^{\mathrm{AP}}$$

Eq 2. Response time of Hi-tasks from Mid- to Hi- mode (Lo-tasks are terminated)

Theoretical Model and Optimisation

- To support the newly introduced Mid-Mode, we need
  - (a) To have a corresponding timing analysis;
  - (b) A way to work out the best switching instances.
- For (b), we proposed two strategies to find the optimal switching time:
  - 1. Solving by priority ordering: choose the Lo-task that has the highest priority first, then scale its C_i^{MI} until the system is not schedulable.

$$C_i^{\mathrm{MI}} = \arg\max_{C_i}(R_j \leq D_j, \forall j \in \Gamma)$$

  - 2. Solving by global scaling: in this case, the C_i^{MI} of the Lo-tasks will be scaled globally by a factor γ. The searching is done using binary search with a time complexity of $O(\log(n))$.

$$\forall i \in \Gamma^{\mathrm{HI}}, C_i^{\mathrm{MI}} = C_i^{\mathrm{LO}} + \lceil \gamma(C_i^{\mathrm{HI}} - C_i^{\mathrm{LO}}) \rceil$$

Outline

- Mixed-Criticality System (MCS)
- Motivation and Research Challenges: Imprecise MCS
- HIART-MCS: Method, Architecture, and Design
- Theoretical Model and Optimisation
- **Evaluation**
- Conclusion

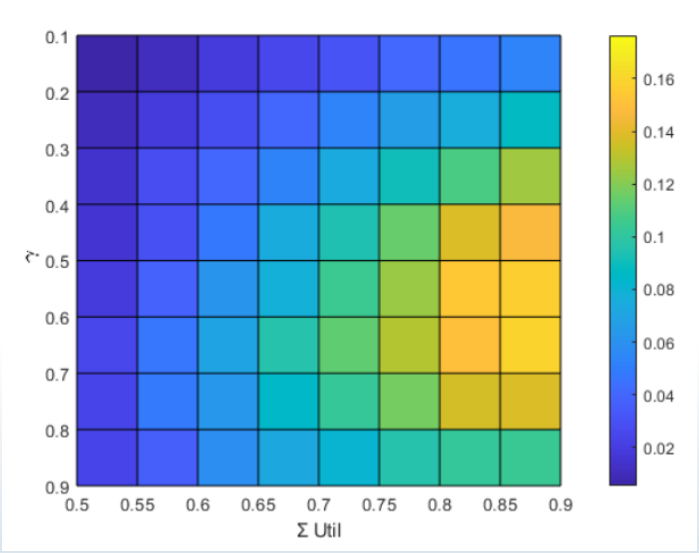
Experimental Platform

- Platform: Xilinx VC709 Evaluation Board
- Processors: 16 HIART-processor
  - RSIC-V ISA
  - 5-stage pipeline
  - 4KB instruction and data cache
- Interconnect: 5 x 5 BlueShell Network-on-Chip
- Operating system: FreeRTOS v.10.4

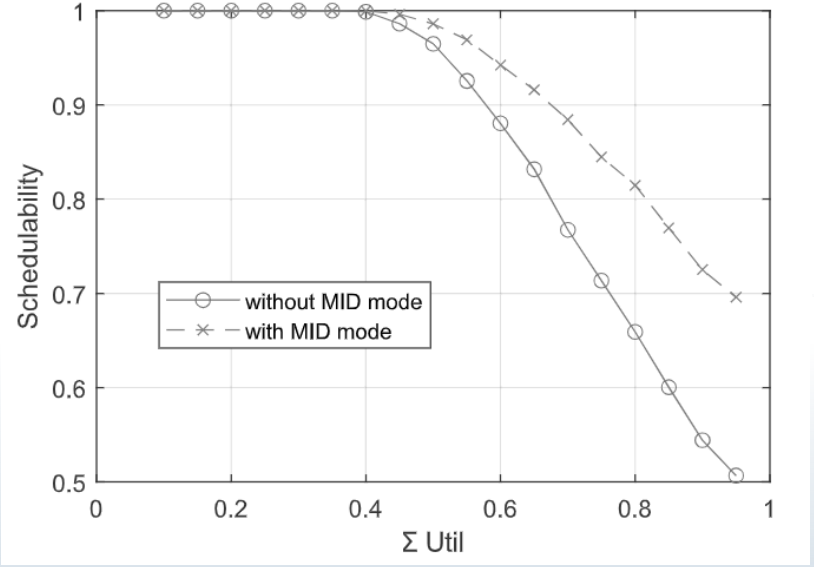
- Examined systems
  - Legacy: a system without MCS feature
  - BS|OSK: an MCS running execution monitor inside OS kernel
  - BS|HYP : an MCS running execution monitor interpedently
  - HIART-MCS

Theoretical Evaluation

- Experiment Setup:
  - Tasks are randomly generated, with utilization ranging from 0.5-0.95 (0.1-0.9 for the schedulability experiment); γ is set to be 0.1-0.9 (and in the schedulability analysis, it was searched using the proposed method).
  - Each trial (a single data point in the plots) consist 1,000 runs.
- Observation: The proposed MCS model outperforms the traditional dual-mode model (i.e. without the Mid-mode) in terms of survivability and schedulability.



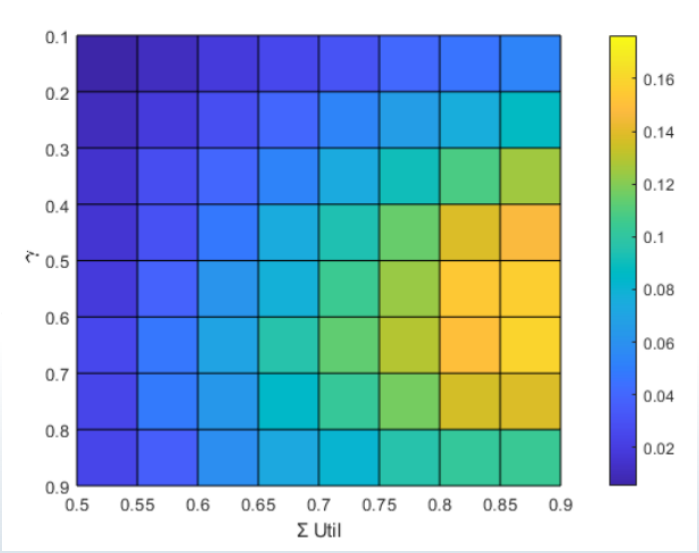
(a) Survivability evaluation (in difference)



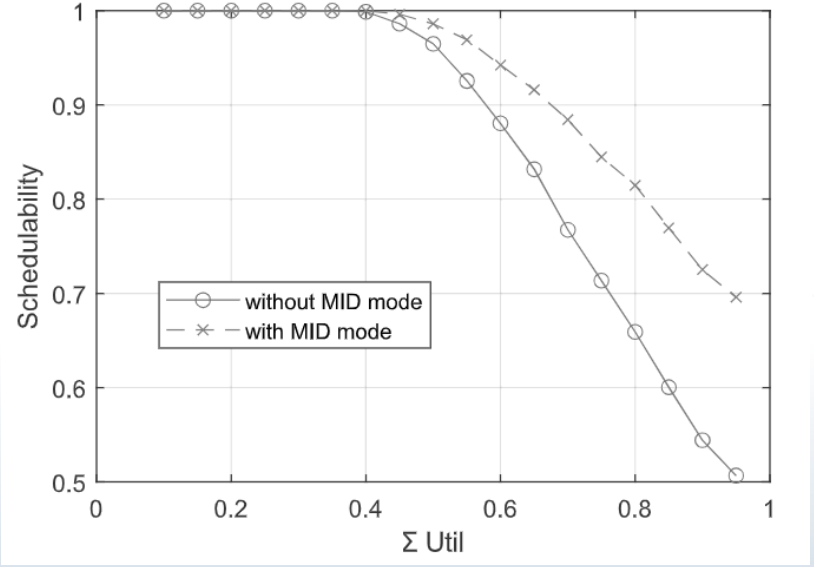
(b) Schedulability evaluation

Theoretical Evaluation

- Experiment Setup:
  - Tasks are randomly generated, with utilization ranging from 0.5-0.95 (0.1-0.9 for the schedulability experiment); γ is set to be 0.1-0.9 (and in the schedulability analysis, it was searched using the proposed method).
  - Each trial (a single data point in the plots) consist 1,000 runs.
- Observation: The proposed MCS model outperforms the traditional dual-mode model (i.e. without the Mid-mode) in terms of survivability and schedulability.



(a) Survivability evaluation (in difference)

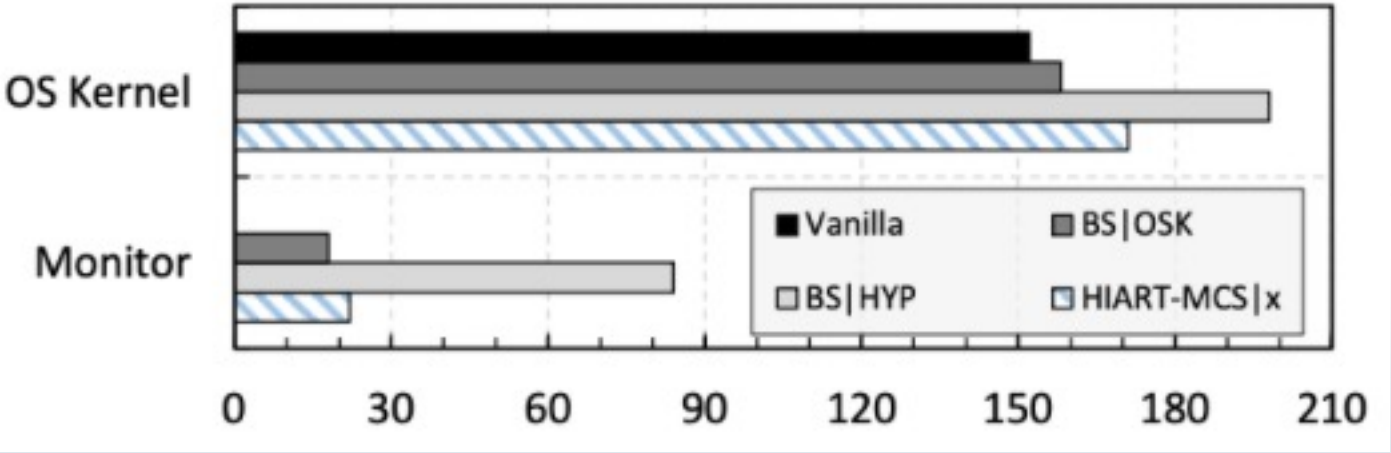


(b) Schedulability evaluation

Software Overhead

- Experimental Setup
  - Operating Systems: native FreeRTOS kernel with essential I/O drivers
  - Tool: RISC-V GNU tool-chain
- Metrics: memory footprint (unit: KB)

- Observation: HIART-MCS requires less memory footprint than BS|HYP. Its software overhead is similar to BS-OSK.



Run-time Software Overhead (Unit: KB)

Hardware Overhead

- Experimental Setup
  - HIART-Processor: support run-time approximation of 100 tasks
  - Conventional RSIC-V processor
  - Other system elements: AXI-Interconnect, SPI controller, and Ethernet controller
  - Tool: Xilinx Vivado (v2020.2)
- Metrics: LUTs, registers, DSP, RAM, and Power.

- Observation: The design of the HIART-processor is resource-efficient compared to a generic processor. The introduced overhead is less than the basic system elements.

	LUTS	Registers	DSP	RAM(KB)	Power(mW)
AXI-IC	543	724	0	0	11
SPI	695	512	0	0	7
Ethernet	1428	895	0	0	13
RISC-V Processor	4,993	5,322	17	256	377
HIART-Processor	5,329	5,743	17	256	389
Proposed	216	387	0	0	5

Hardware Overhead

Case Study

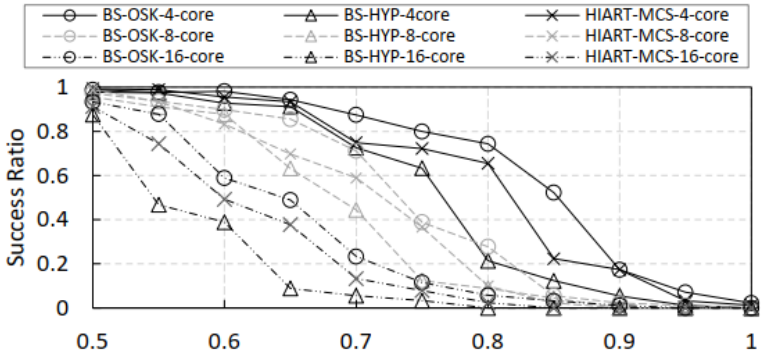
- Task sets
  - 18 Hi-tasks
    - Renesas functional safety automotive use cases
  - 18 Lo-tasks
    - EEMBC benchmark
    - DNN tasks based on LeNet-5 and SqueezeNet
    - Image processing tasks: Sobel, Canny, Scharr, Prewitt, Roberts, Sharpen filters

- Experimental setup
  - Activating 4/8/16 processors
  - Tunning system *target utilisation* from 45% to 100%

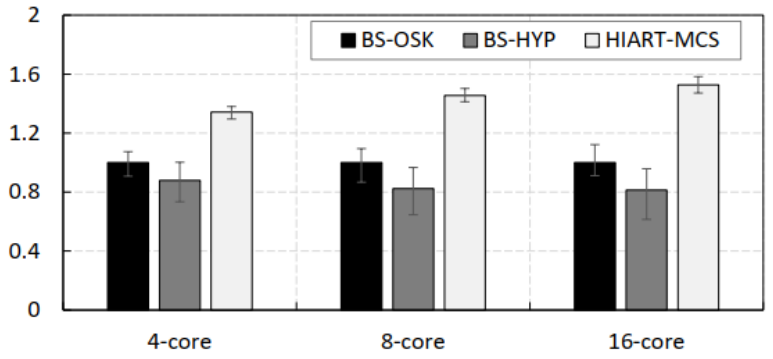
- Metrics
  - Hi-tasks: success ratio
  - Lo-Tasks: number of services (NoS)
  - Hi-Tasks and Lo-Tasks: average Quality of Computation (QoC)

Case Study: Results

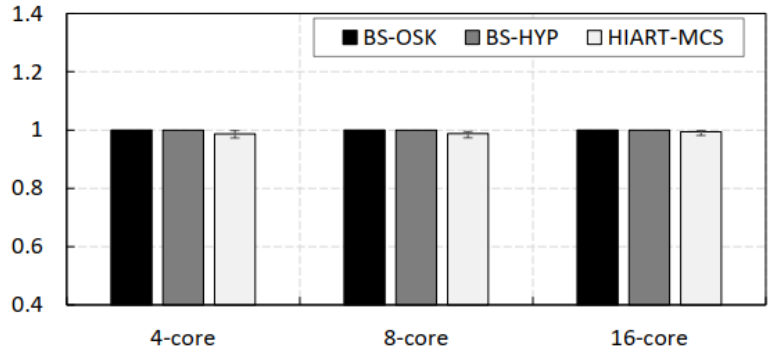
- Observations
  - For HI-tasks, HIART-MCS ensures similar success ratios compared to conventional MCS frameworks
  - For LO-tasks, HIART-MCS significantly increases throughput and decreases experimental variances compared to conventional MCS frameworks
  - HIART-MCS slightly decreases overall computation quality



(a) HI-tasks: success ratio (x-axis: utilisation).



(b) LO-tasks: average NoS.



(c) System level: average QoC.

Outline

- Mixed-Criticality System (MCS)
- Motivation and Research Challenges: Imprecise MCS
- HIART-MCS: Method, Architecture, and Design
- Theoretical Model and Optimisation
- Evaluation
- **Conclusion**

Conclusion

- In safety-critical systems, Mixed-Criticality System (MCS) is a vital direction.
- Mode switch is a useful strategy in MCS but could cause safety risks.
- To mitigate these risks, we present a systematic framework, named HIART-MCS
  - A processor supporting hardware-level approximation
  - A measurement-based method to configure the processor
  - A new system architecture
  - Theoretical analysis and optimization
- Results
  - Effectively improving LO-task survivability with negligible impact on HI-tasks
  - Resource-efficient

Acknowledgement

We would like to thank the helpful feedback given by the reviewers from RTSS 2021 and the Special issue of real-time systems from Transactions on Computers (TC).