

Distributed run-time WCET controller for concurrent critical tasks in mixed-critical systems

Angeliki Kritikakou^{†‡}, Claire Pagetti[†], Christine Rochange[†],
Matthieu Roy^{*}, Madeleine Faugere , Sylvain Girbal and Daniel Gracia Perez

09/10/2014, RTNS'14, Verseilles, France



Outline

- Introduction & Motivation
- Overview
- Design time analysis
- Run-time control
- Evaluation results
- Conclusions & Future directions

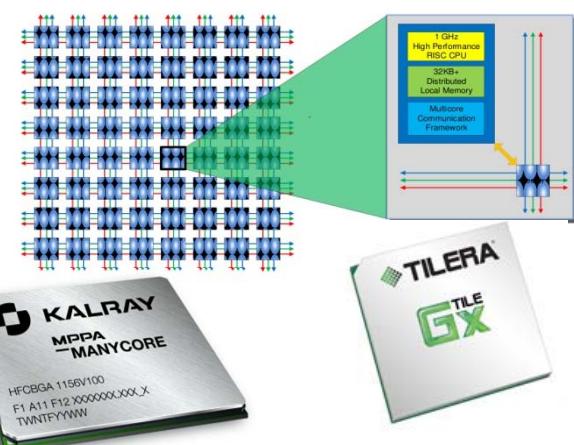
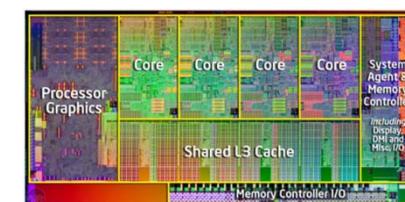
Target domain

- **Software:** Mixed Critical applications
 - Hard & soft deadlines
 - Different consequences in failure
 - Criticality Levels: DAL, ASIL, ...
 - Domains: Aeronautics, Automotive, ...



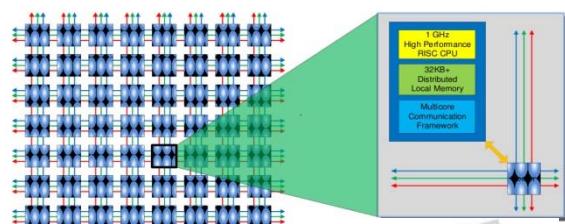
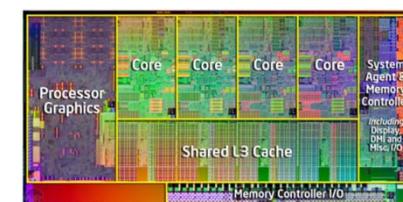
Target domain

- **Software:** Mixed Critical applications
 - Hard & soft deadlines
 - Different consequences in failure
 - Criticality Levels: DAL, ASIL, ...
 - Domains: Aeronautics, Automotive, ...
- **Hardware:** Multi/many-cores
 - Cores with components with dynamic behavior
 - Shared resources
 - COTS: Tilera, Kalray, Texas TMS, ...



Target domain

- **Software:** Mixed Critical applications
 - Hard & soft deadlines
 - Different consequences in failure
 - Criticality Levels: DAL, ASIL, ...
 - Domains: Aeronautics, Automotive, ...
- **Hardware:** Multi/many-cores
 - Cores with components with dynamic behavior
 - Shared resources
 - COTS: Tilera, Kalray, Texas TMS, ...

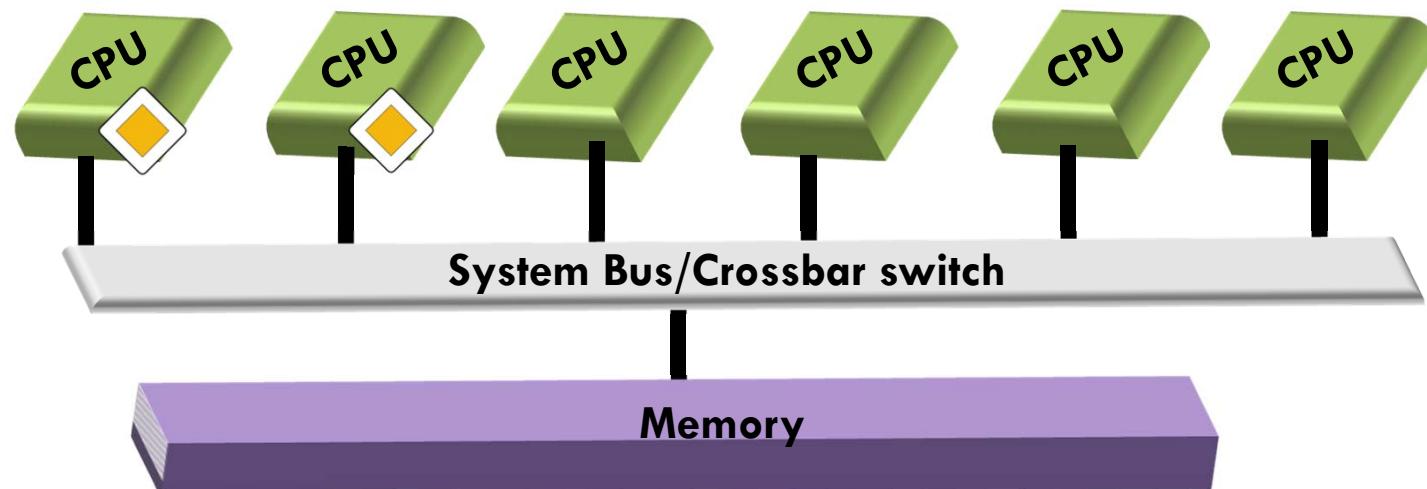


Challenge: Guarantee hard real-time response & improve cores utilization



Motivation

- Safe WCET static analysis for each critical task

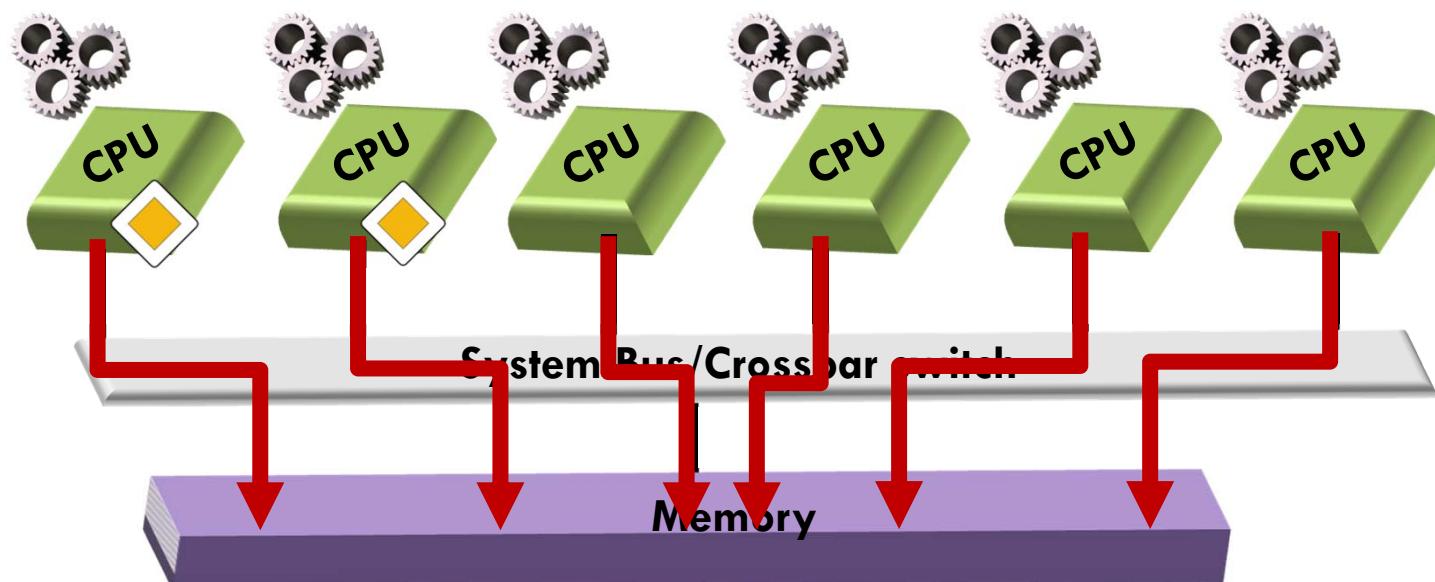


Motivation

- Safe WCET static analysis for each critical task

Many active cores

- Maximum load

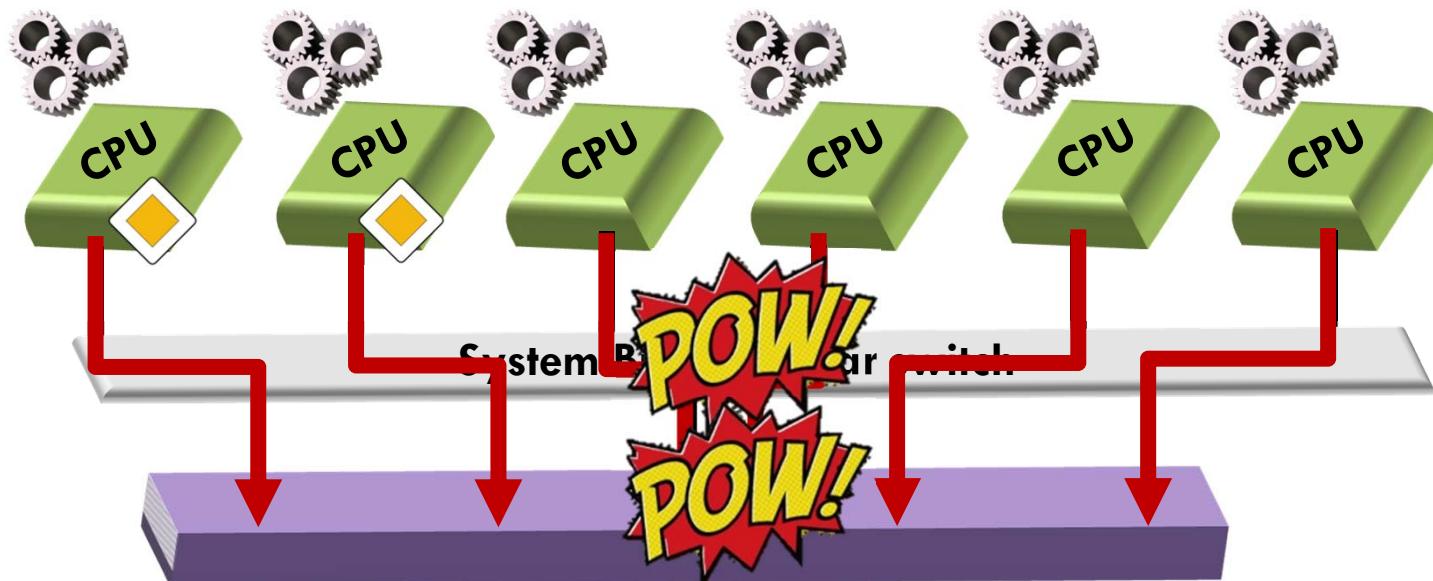


Motivation

- Safe WCET static analysis for each critical task

Many active cores

- Maximum load
- Full congestion

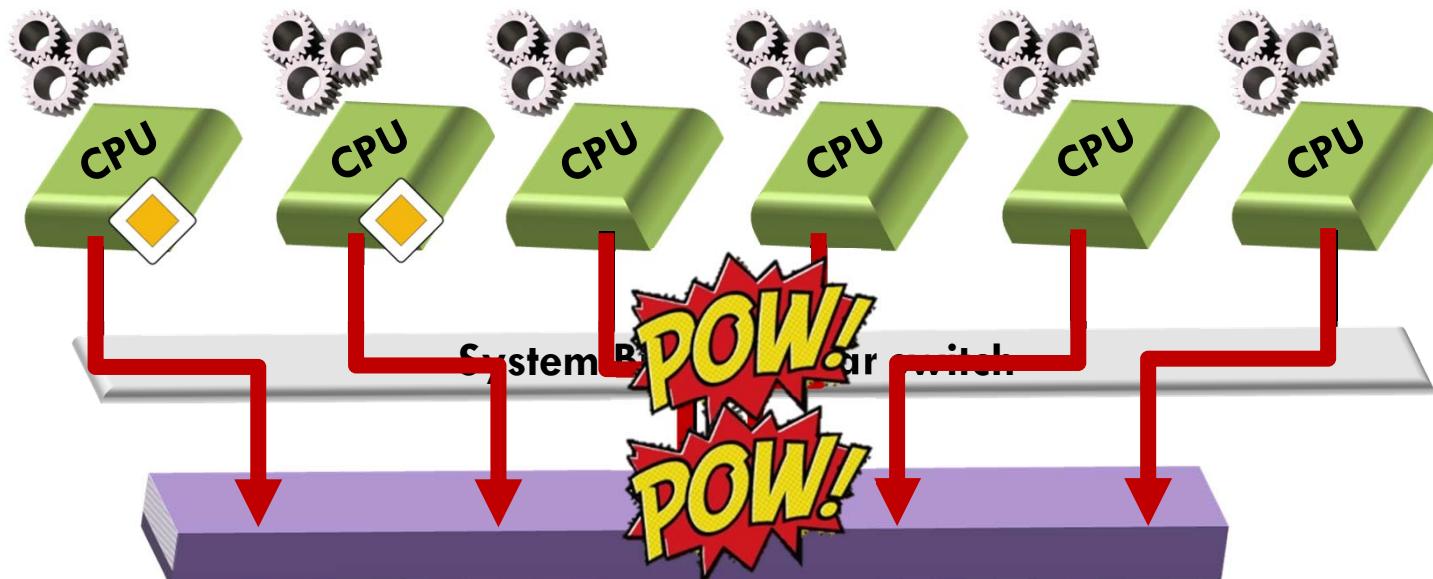


Motivation

- Safe WCET static analysis for each critical task

Many active cores

- Maximum load
- Full congestion



WCET > D_c

Motivation

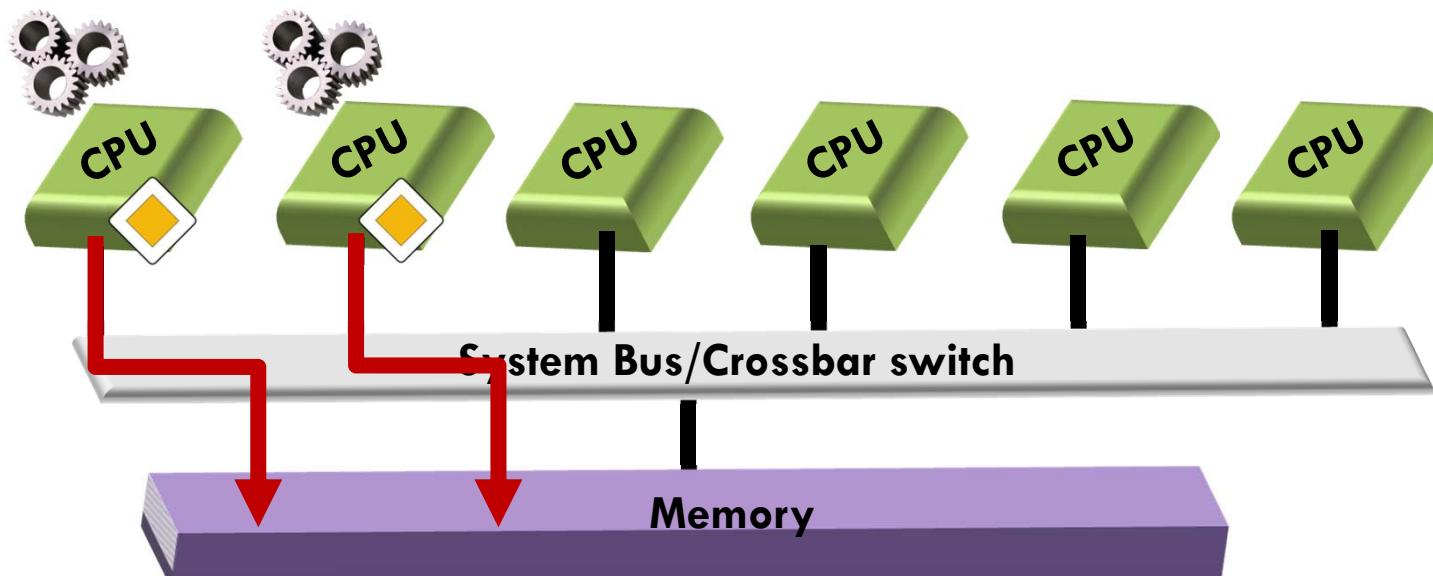
- Safe WCET static analysis for each critical task

Many active cores

- Maximum load
- Full congestion

Few active cores

- Isolation



WCET > D_c

Motivation

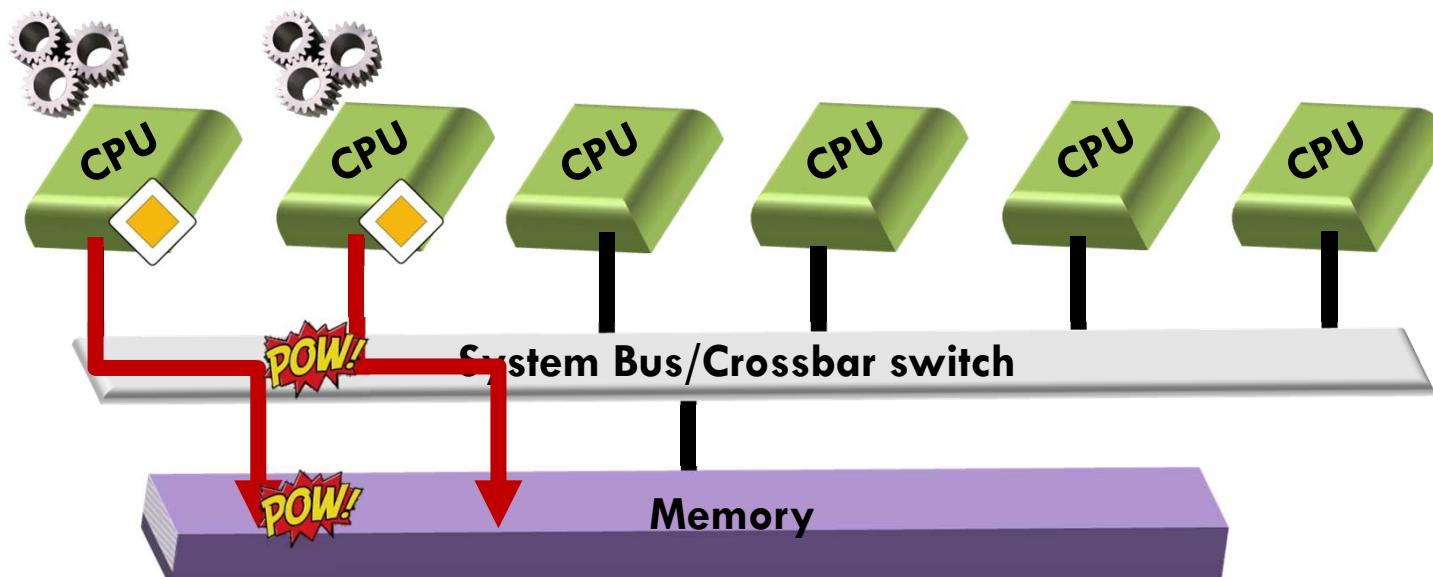
- Safe WCET static analysis for each critical task

Many active cores

- Maximum load
- Full congestion

Few active cores

- Isolation
- Congestion of critical tasks



WCET > D_c

Motivation

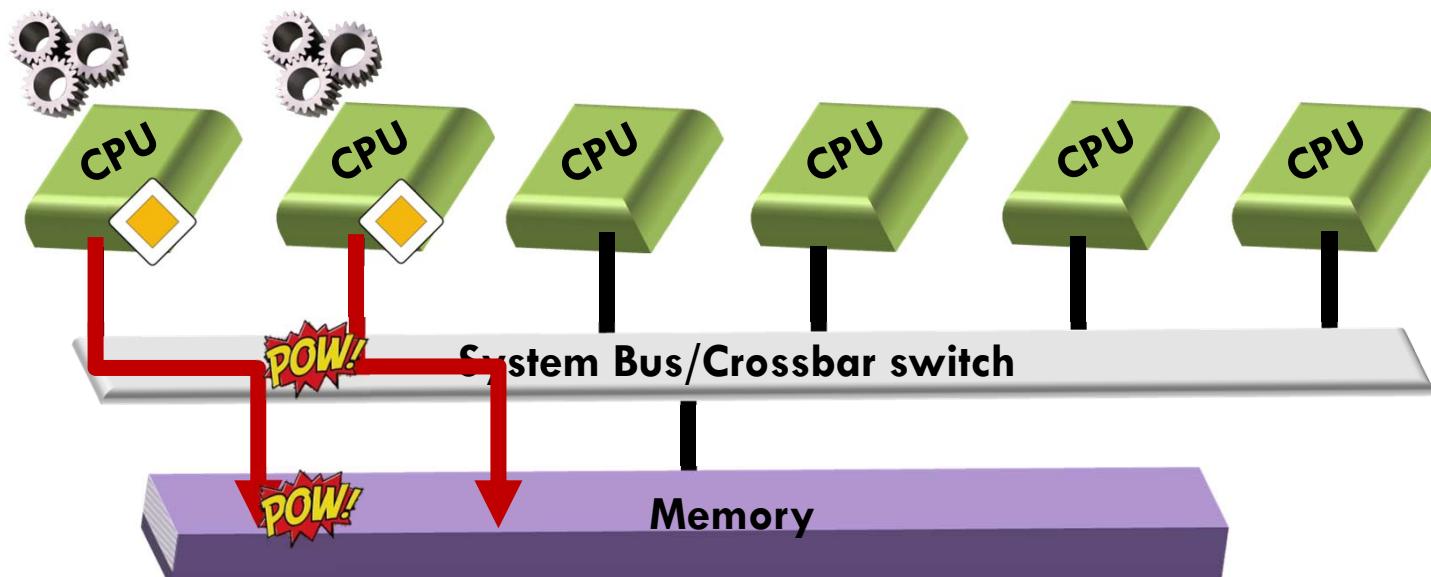
- Safe WCET static analysis for each critical task

Many active cores

- Maximum load
- Full congestion

Few active cores

- Isolation
- Congestion of critical tasks



WCET > D_c

WCET ≤ D_c

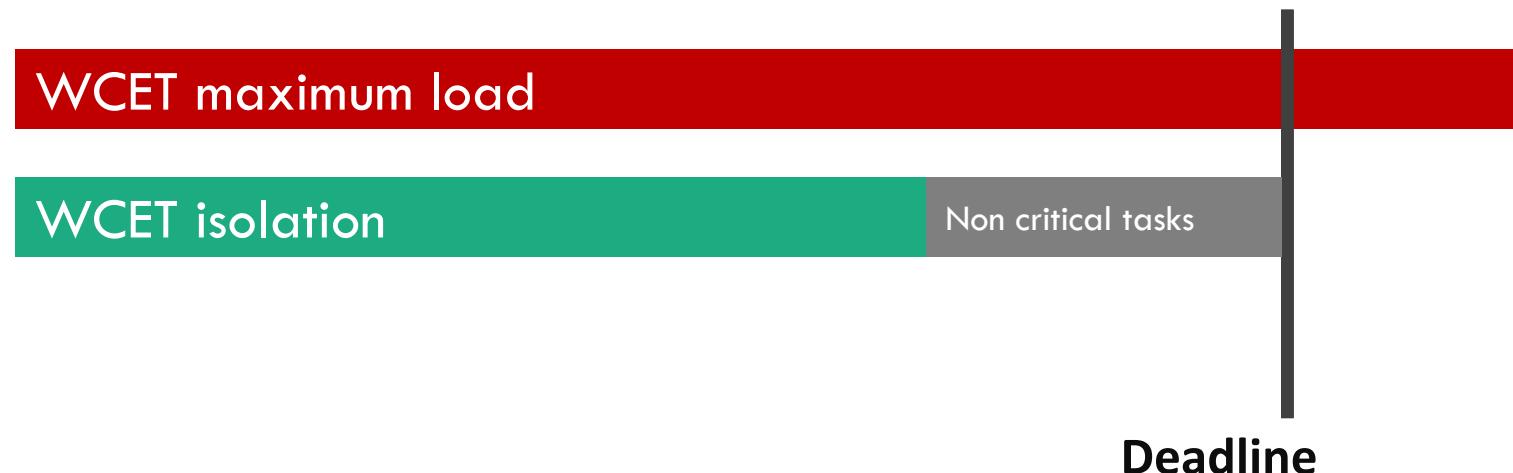
Contribution

WCET maximum load

Deadline

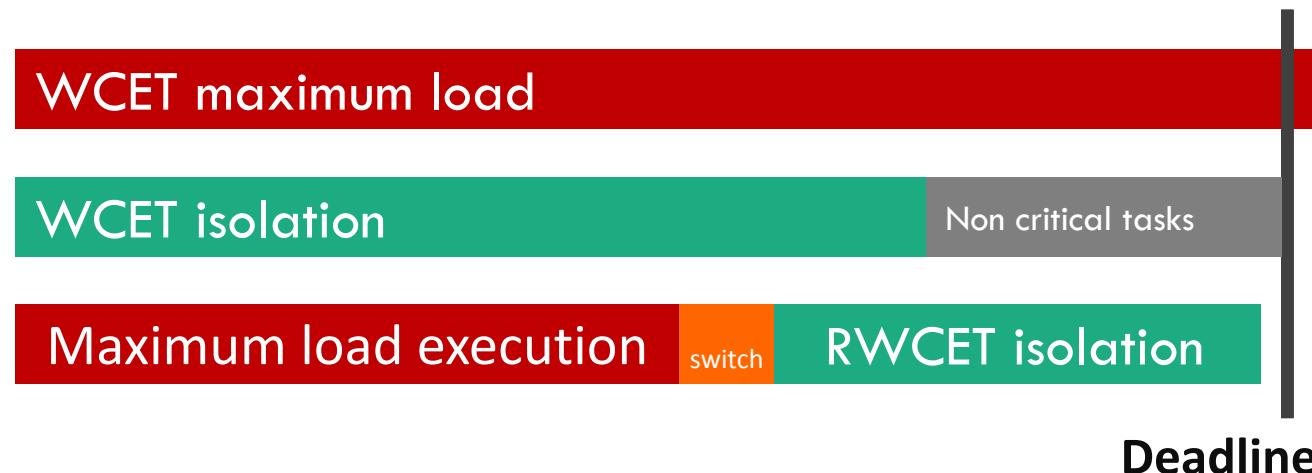
- ✖ Existing approaches: Not directly applicable

Contribution



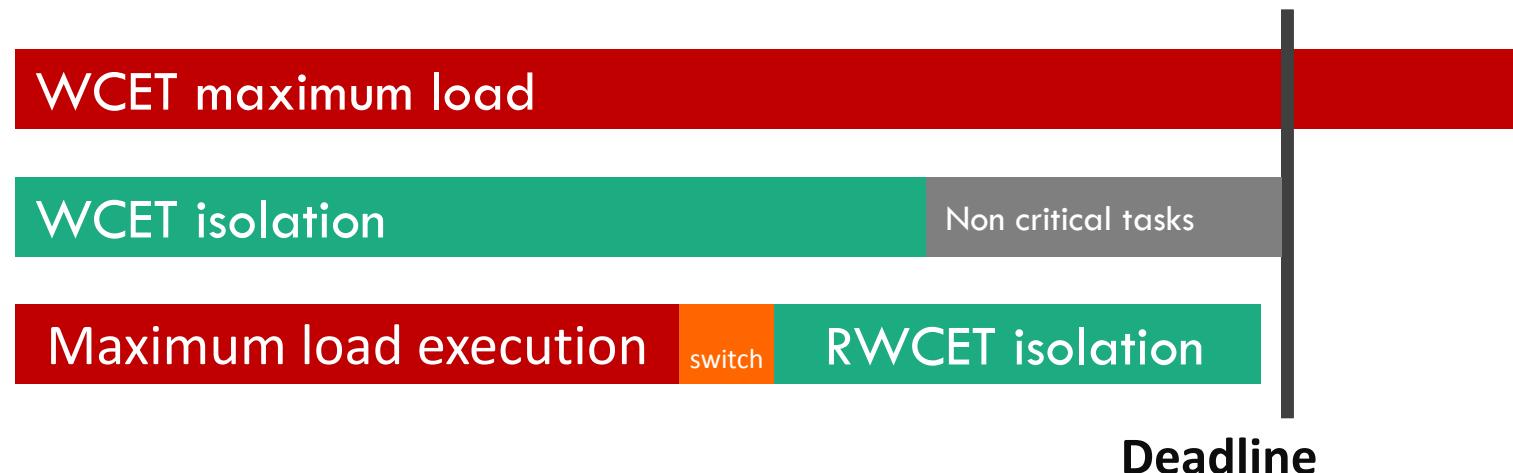
- ✗ Existing approaches: Not directly applicable
- ✓ Safe: Only the most critical tasks
 - ✗ Under-utilization of the system resources

Contribution



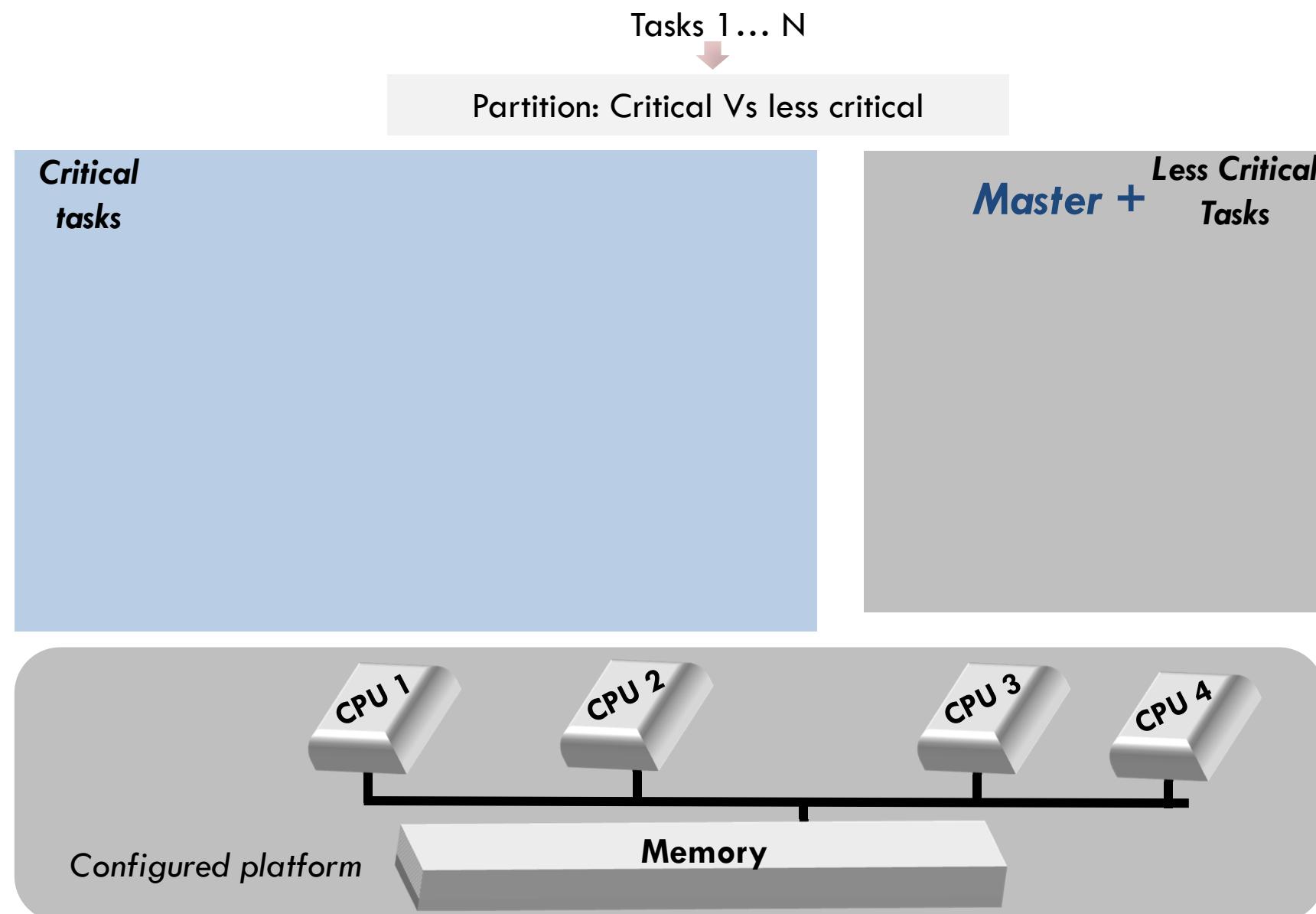
- ✗ Existing approaches: Not directly applicable
- ✓ Safe: Only the most critical tasks
 - ✗ Under-utilization of the system resources
- ✓ Proposed: Maximum load if safe, ELSE switch to isolation
 - ✓ Improved system resources utilization
 - ✓ Guarantee critical deadlines

Contribution

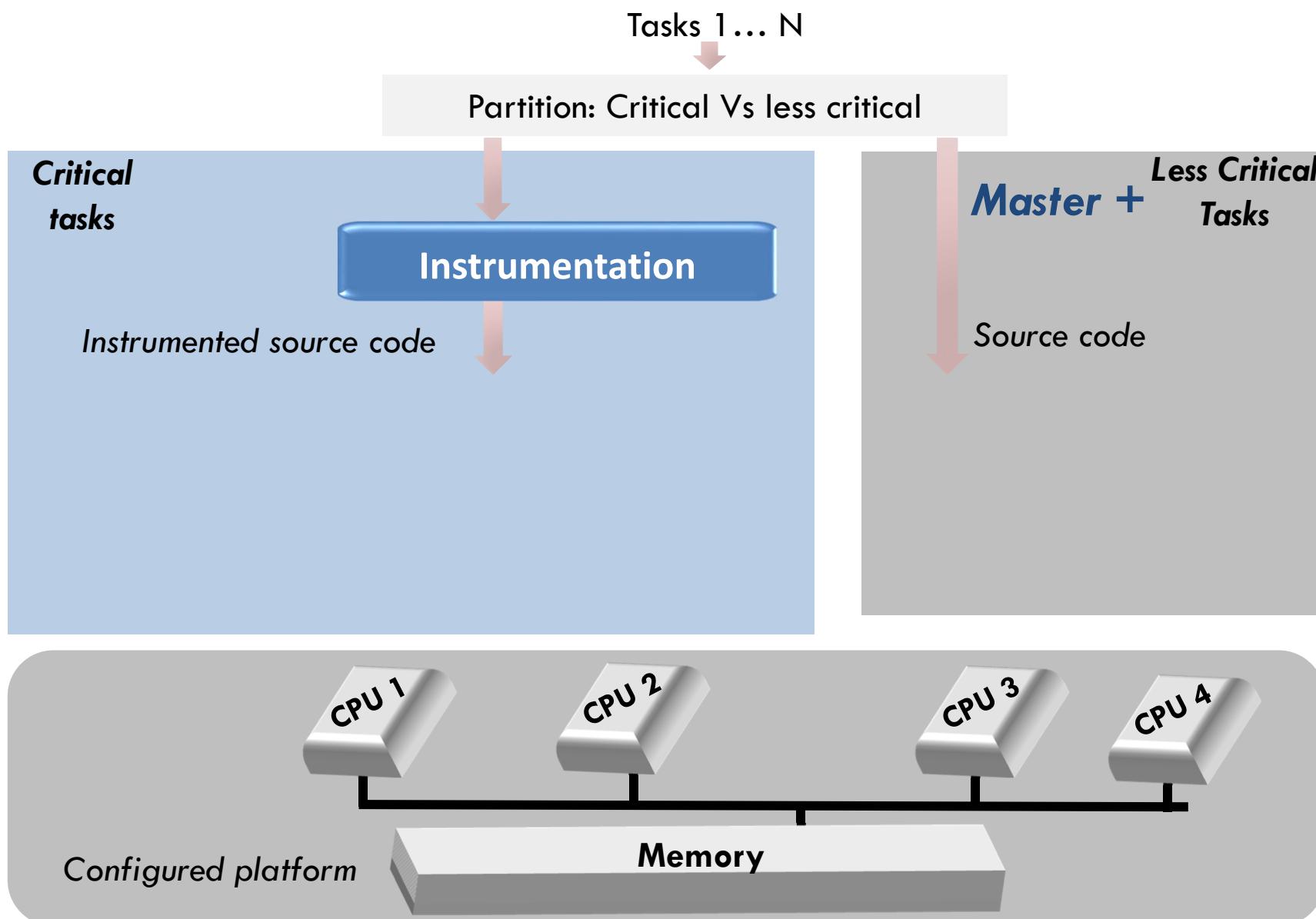


- ✗ Existing approaches: Not directly applicable
 - ✓ Safe: Only the most critical tasks
 - ✗ Under-utilization of the system resources
 - ✓ Proposed: Maximum load if safe, ELSE switch to isolation
 - ✓ Improved system resources utilization
 - ✓ Guarantee critical deadlines
- [ECRTS'14] → Approach to **more than 1** critical task
Implementation on real multicore platform

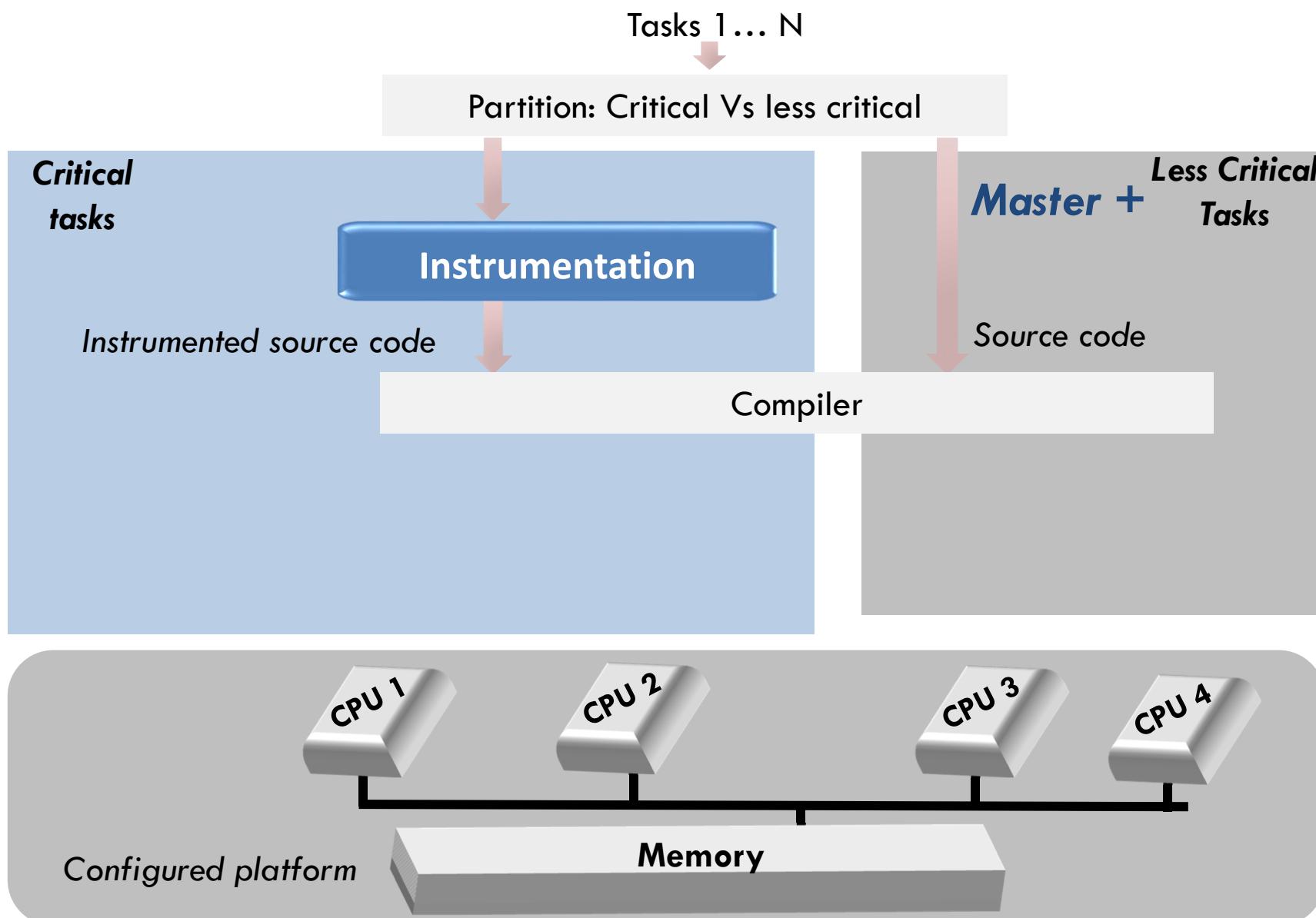
Methodology overview: Design-time



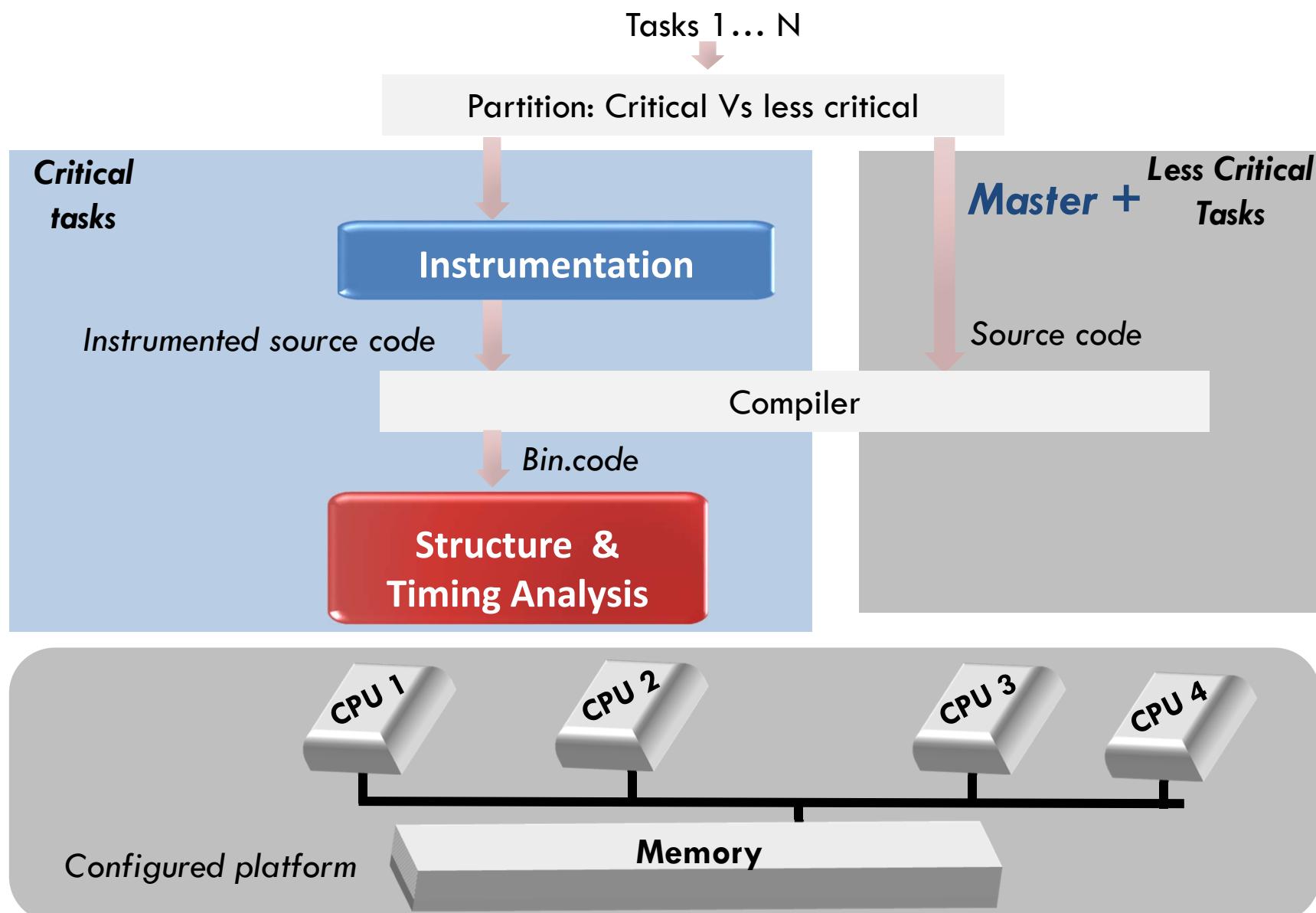
Methodology overview: Design-time



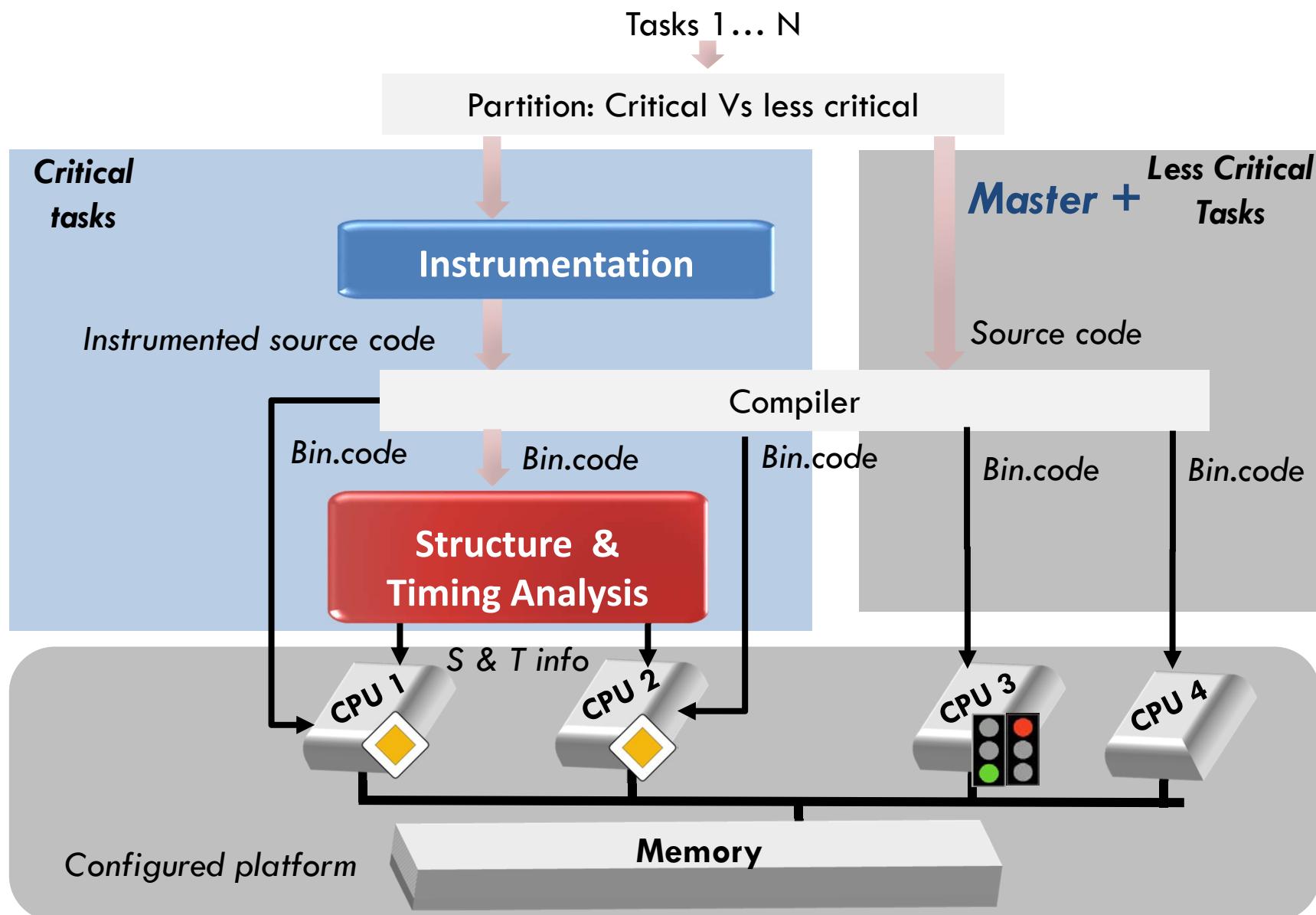
Methodology overview: Design-time



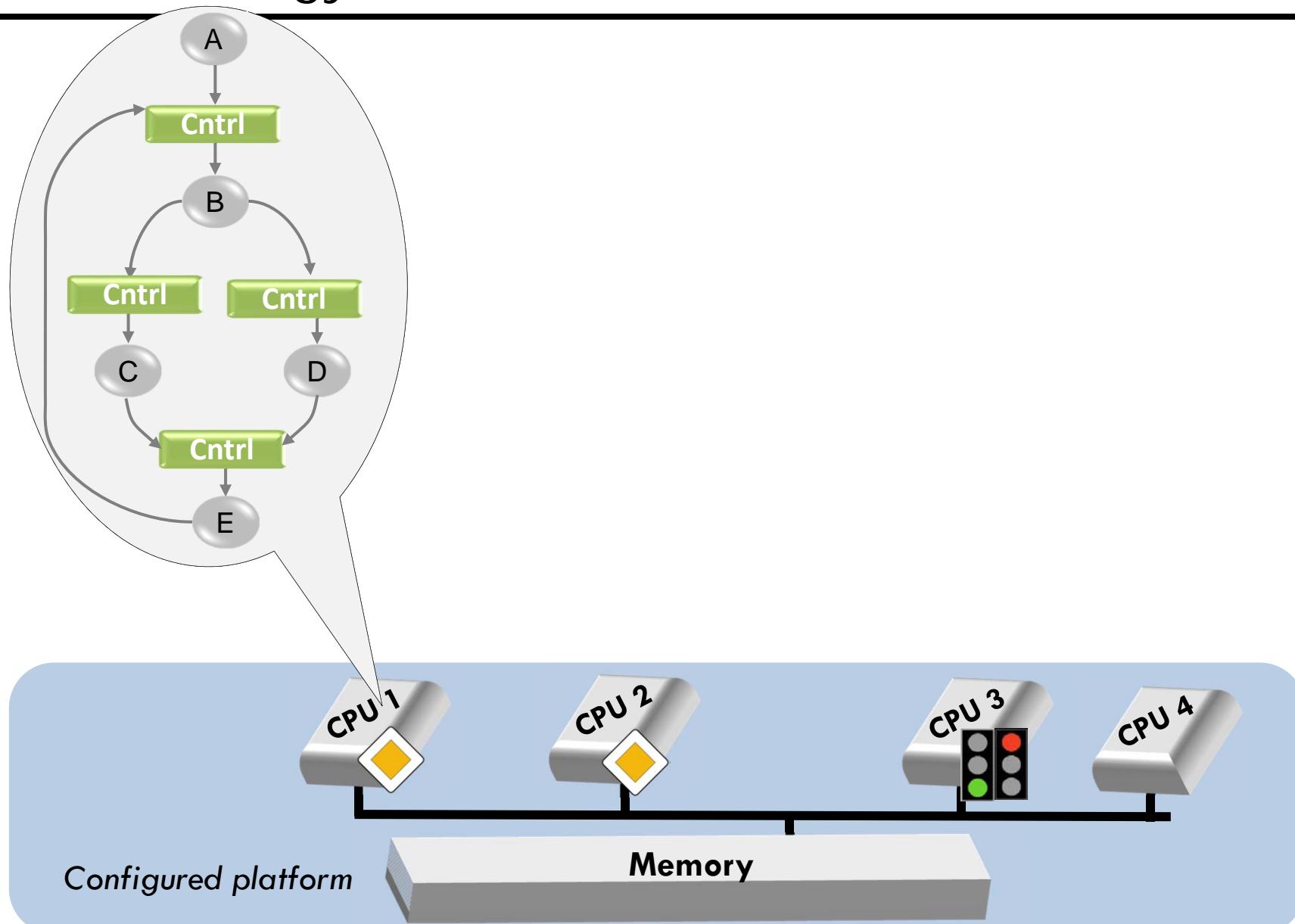
Methodology overview: Design-time



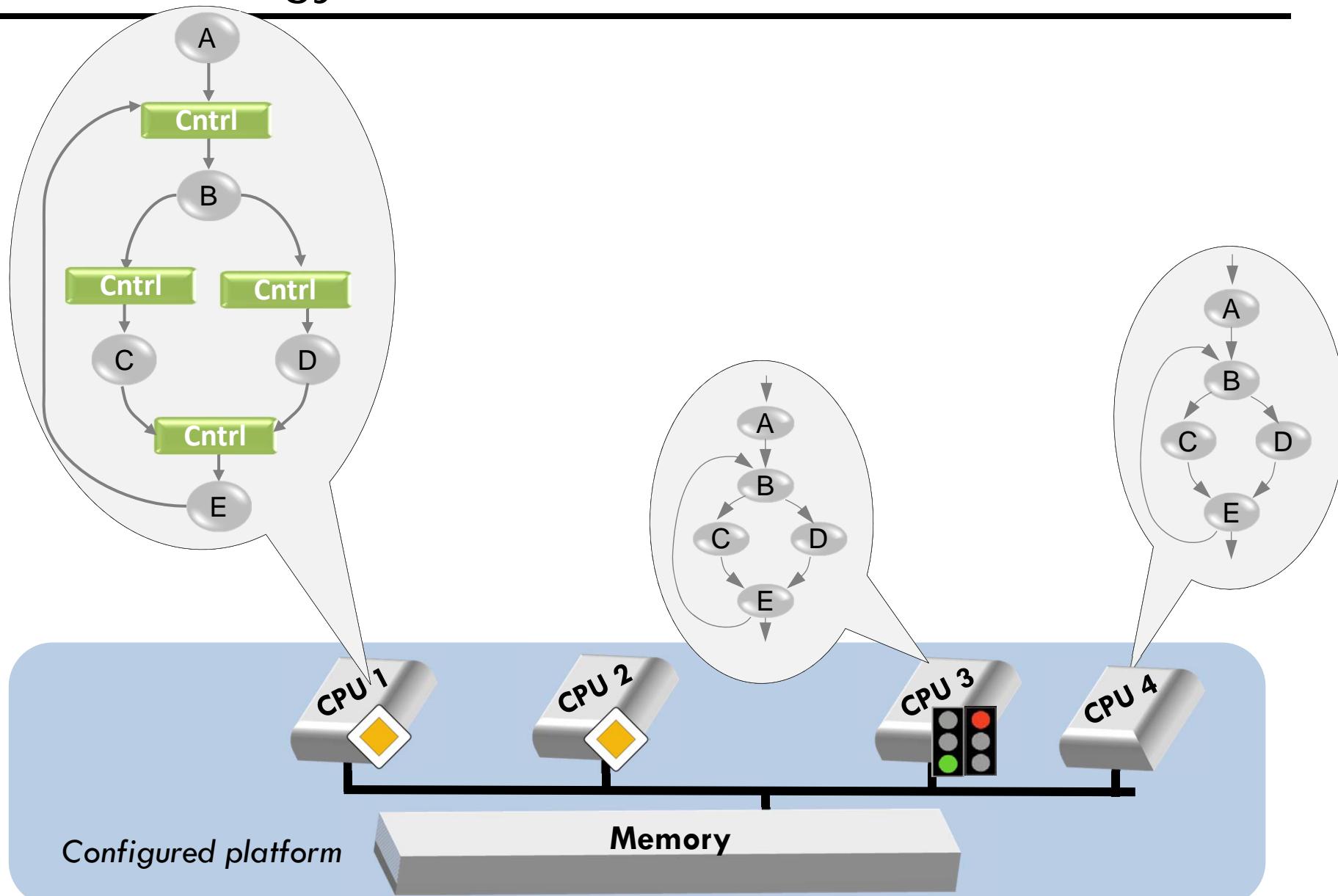
Methodology overview: Design-time



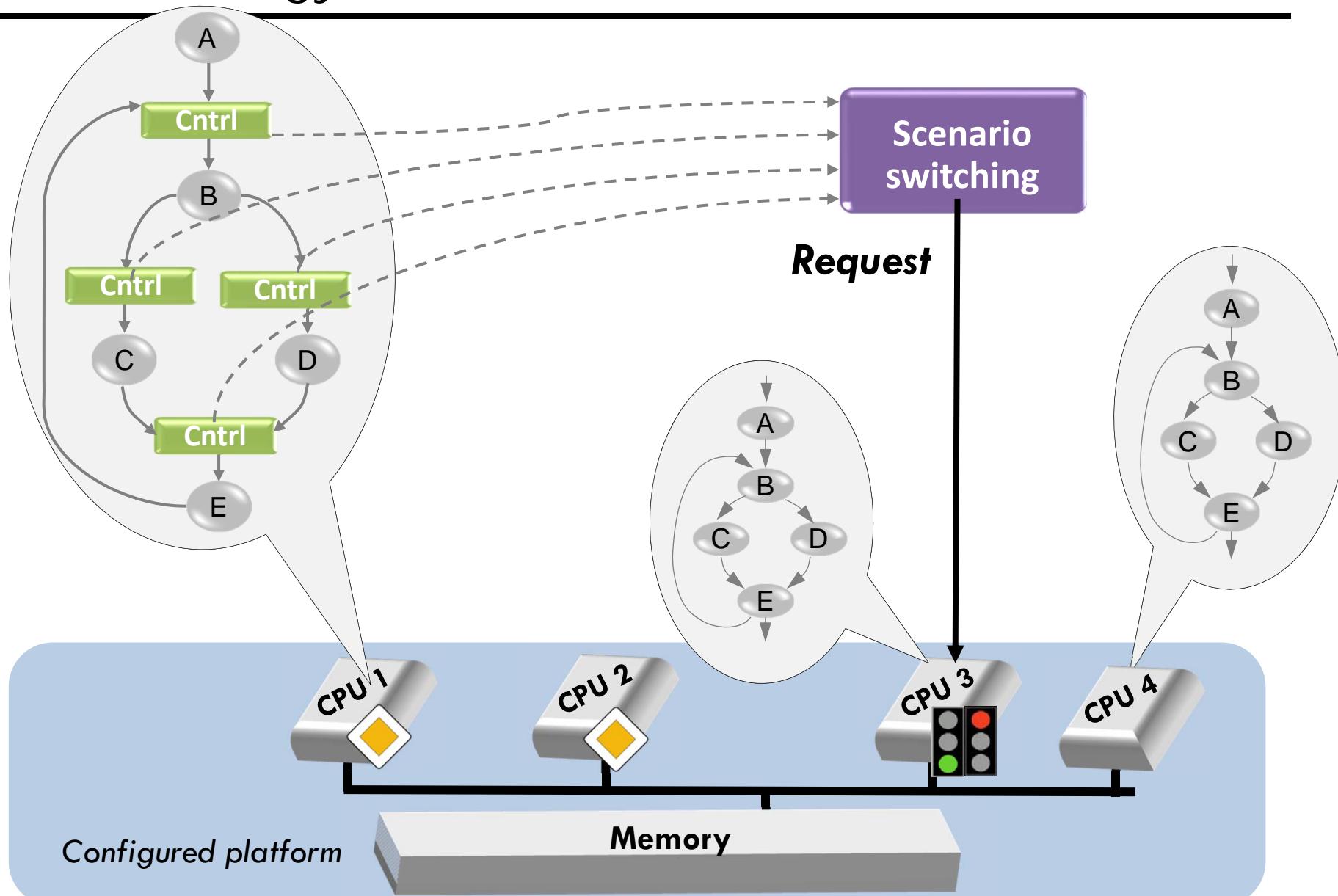
Methodology overview: Run-time



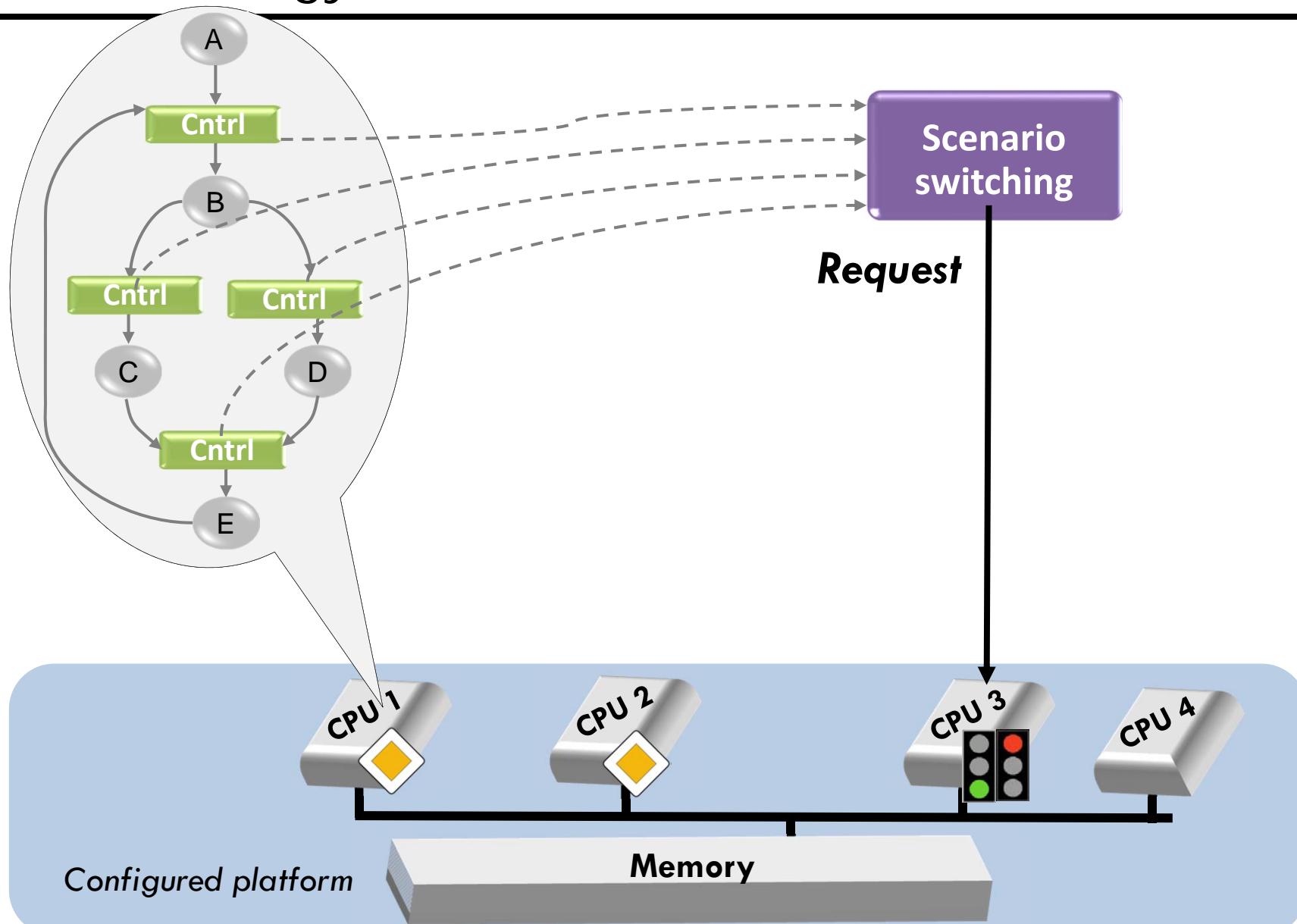
Methodology overview: Run-time



Methodology overview: Run-time



Methodology overview: Run-time



Instrumentation

- Model task as a set of functions: $S = \{F_0, \dots, F_n\}$
 - Each function is an Extended Control Flow Graph (ECFG)



Instrumentation

- Model task as a set of functions: $S = \{F_0, \dots, F_n\}$
 - Each function is an Extended Control Flow Graph (ECFG)



- Observation points per node & at the end of ECFG of F_0

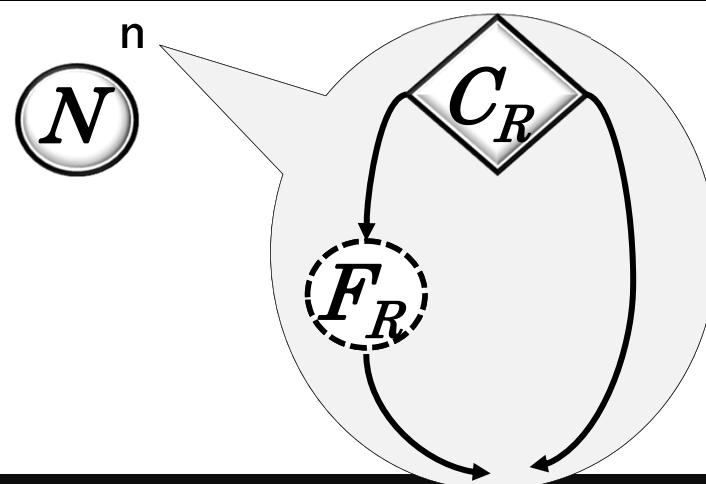
Instrumentation

- Model task as a set of functions: $S = \{F_0, \dots, F_n\}$
 - Each function is an Extended Control Flow Graph (ECFG)



- Observation points per node & at the end of ECFG of F_0

C_R	Node	



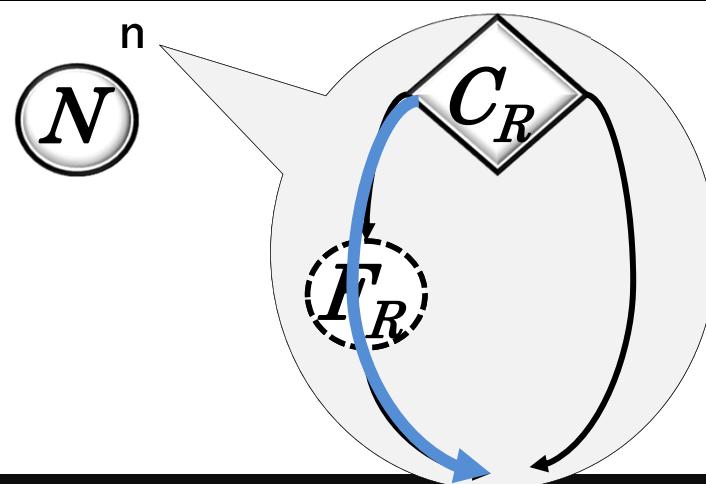
Instrumentation

- Model task as a set of functions: $S = \{F_0, \dots, F_n\}$
 - Each function is an Extended Control Flow Graph (ECFG)



- Observation points per node & at the end of ECFG of F_0

C_R	Node	
1 (Maximum load)	Call F_R	



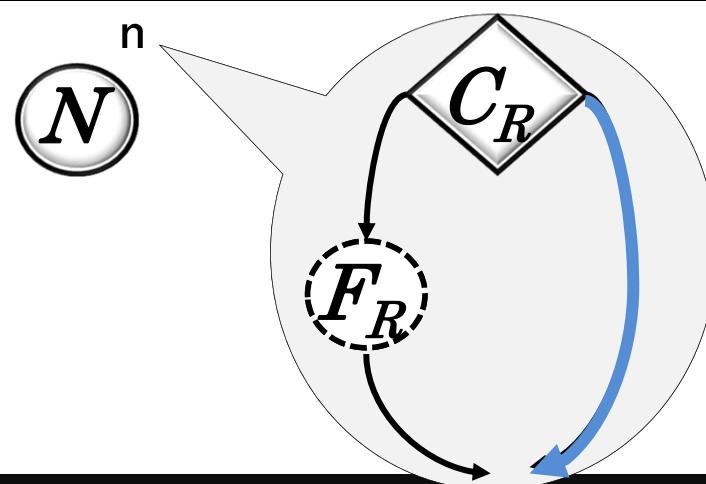
Instrumentation

- Model task as a set of functions: $S = \{F_0, \dots, F_n\}$
 - Each function is an Extended Control Flow Graph (ECFG)



- Observation points per node & at the end of ECFG of F_0

C_R	Node	
1 (Maximum load)	Call F_R	
0 (Isolation)	-	



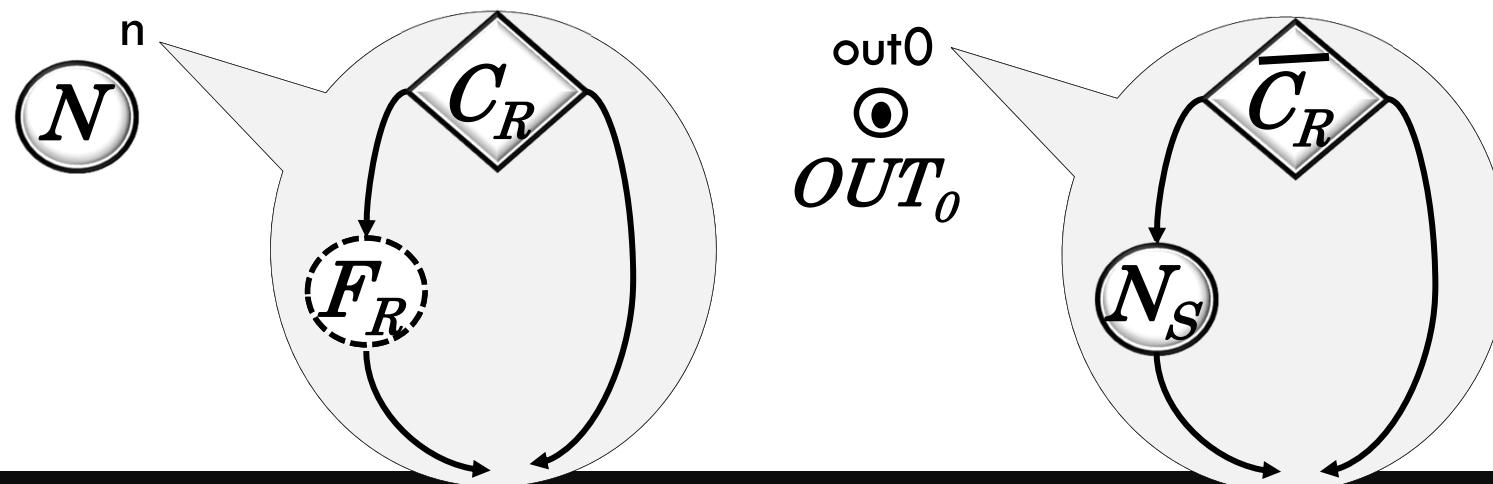
Instrumentation

- Model task as a set of functions: $S = \{F_0, \dots, F_n\}$
 - Each function is an Extended Control Flow Graph (ECFG)



- Observation points per node & at the end of ECFG of F_0

C_R	Node	End of ECFG
1 (Maximum load)	Call F_R	
0 (Isolation)	-	



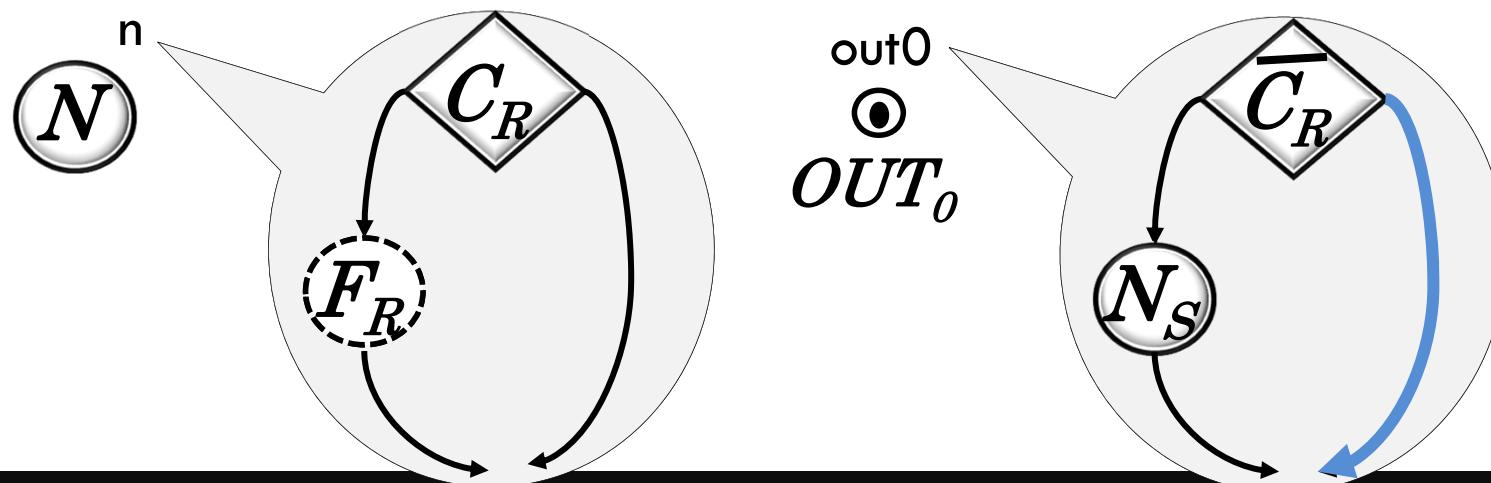
Instrumentation

- Model task as a set of functions: $S = \{F_0, \dots, F_n\}$
 - Each function is an Extended Control Flow Graph (ECFG)



- Observation points per node & at the end of ECFG of F_0

C_R	Node	End of ECFG
1 (Maximum load)	Call F_R	-
0 (Isolation)	-	-



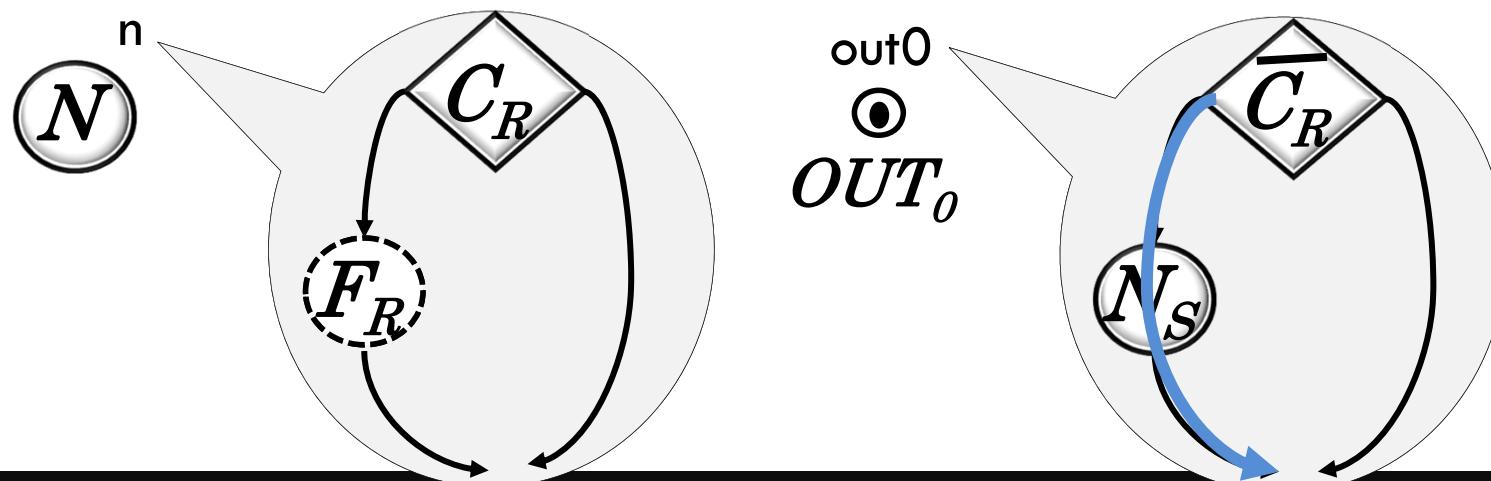
Instrumentation

- Model task as a set of functions: $S = \{F_0, \dots, F_n\}$
 - Each function is an Extended Control Flow Graph (ECFG)

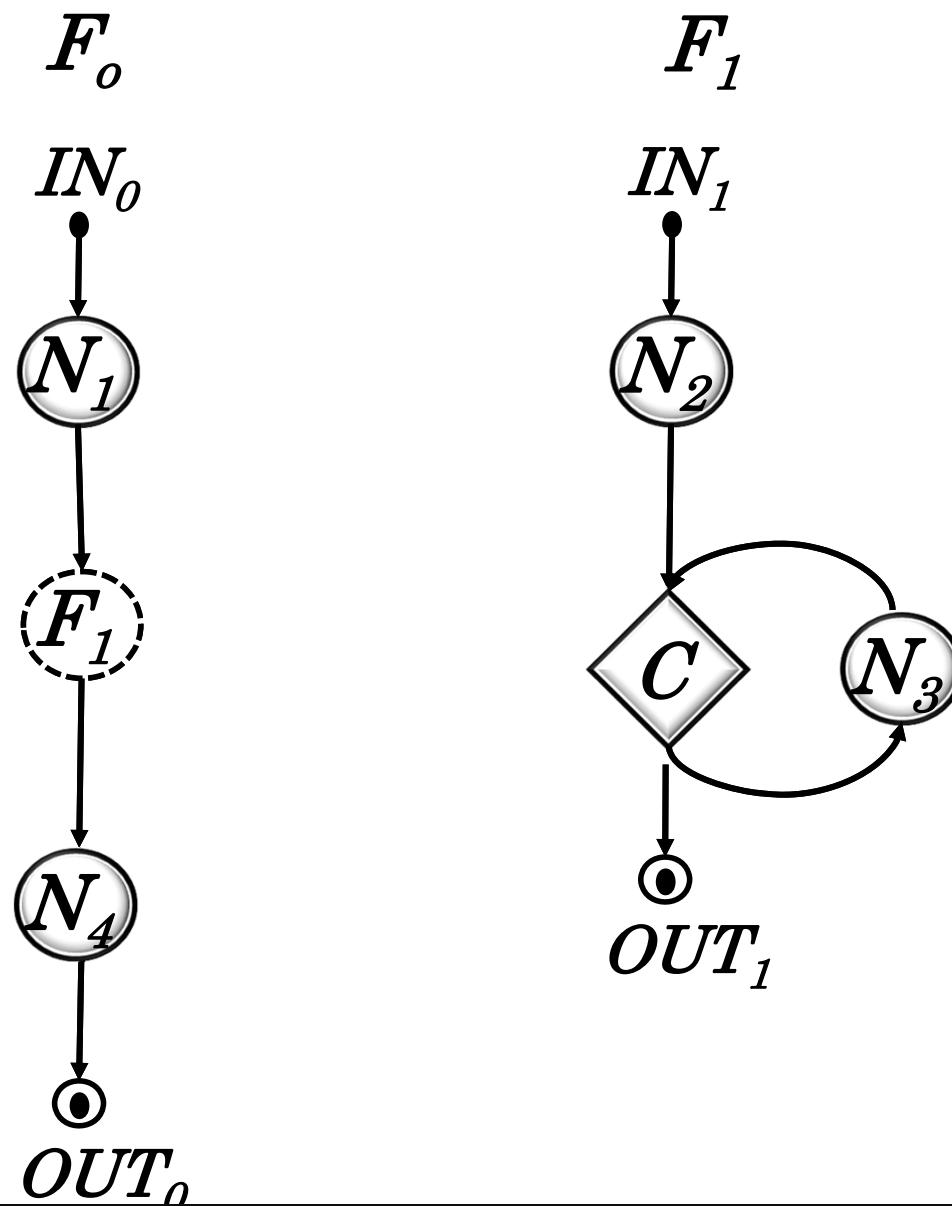


- Observation points per node & at the end of ECFG of F_0

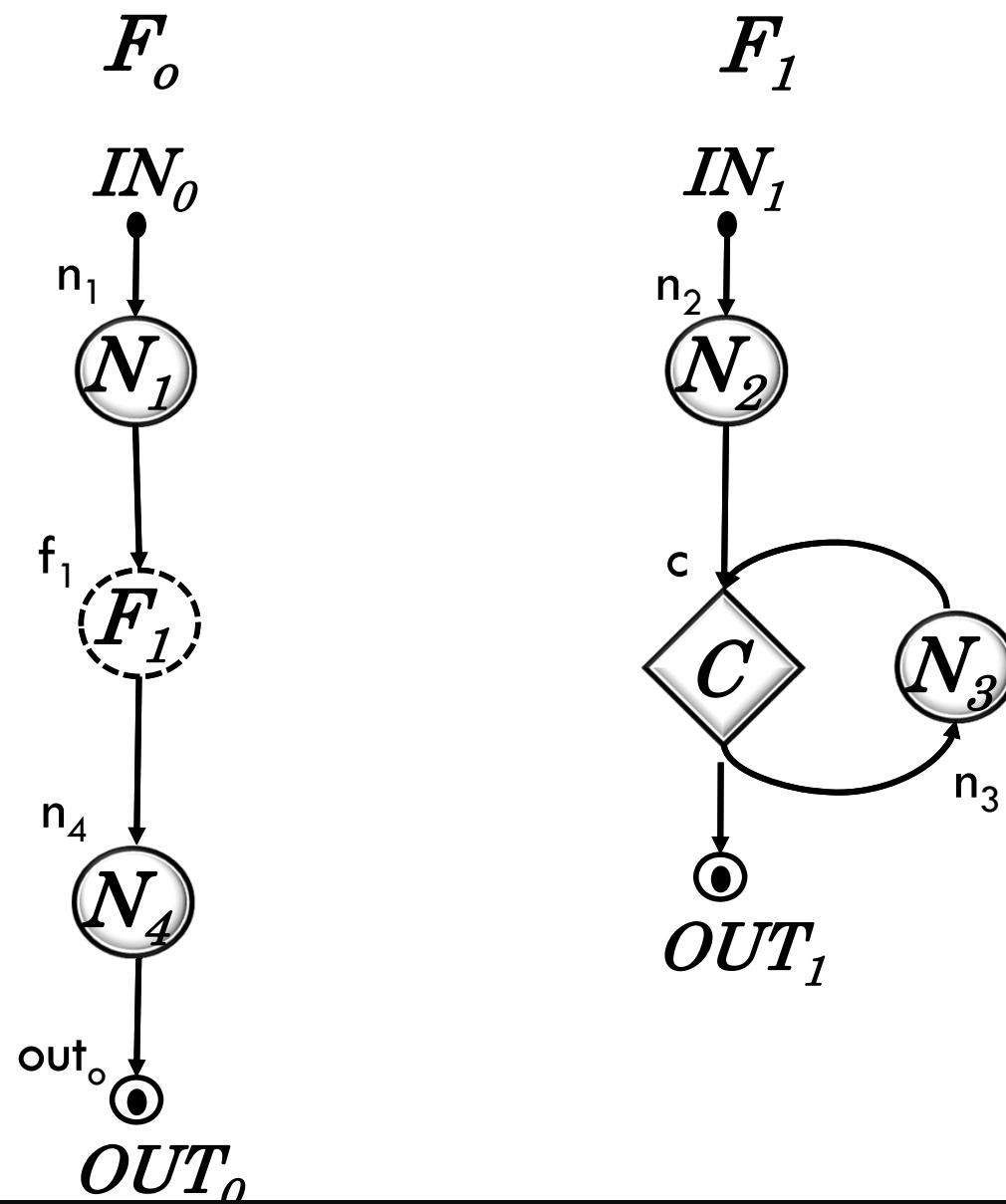
C_R	Node	End of ECFG
1 (Maximum load)	Call F_R	-
0 (Isolation)	-	Notify termination



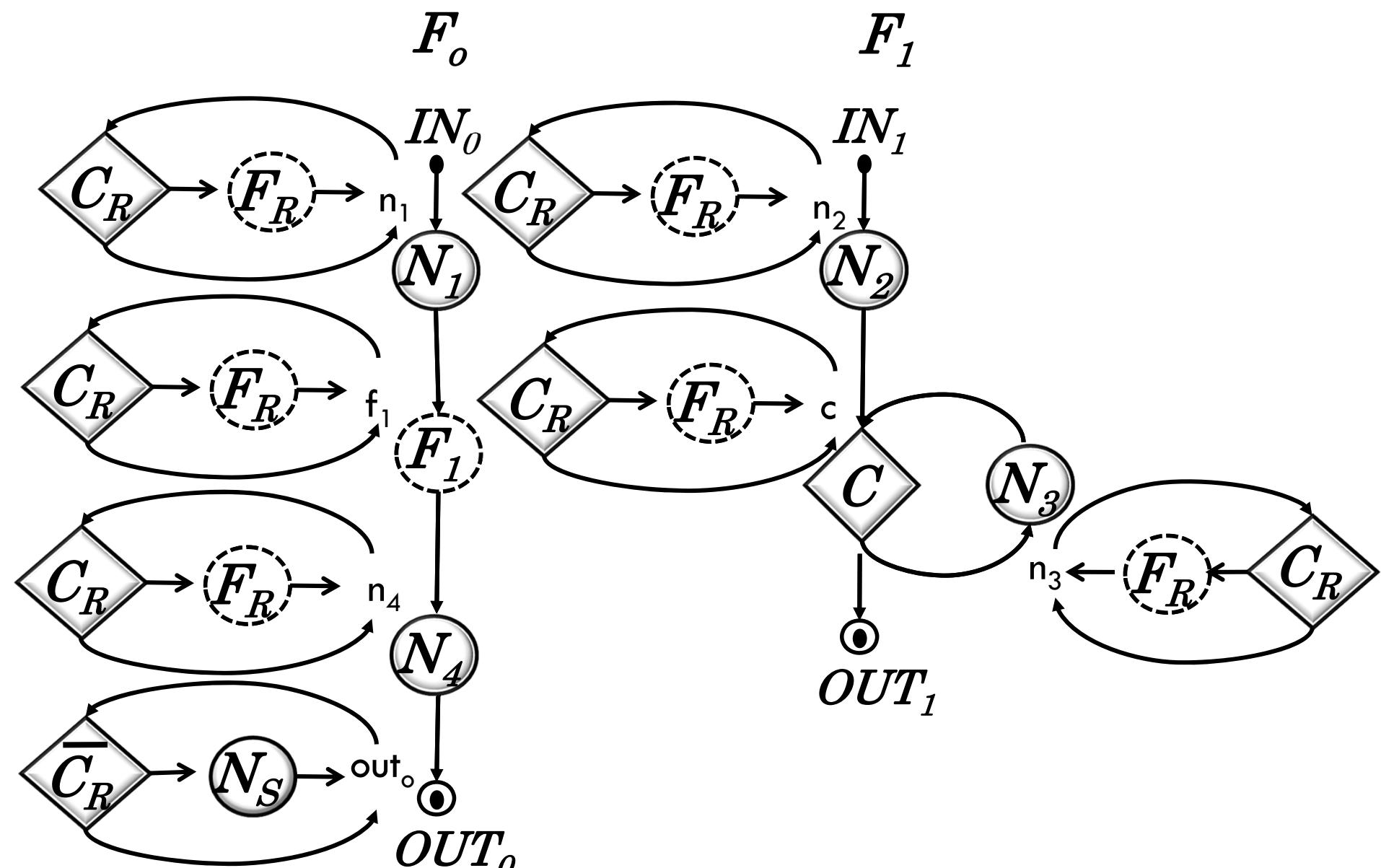
Instrumentation example



Instrumentation example

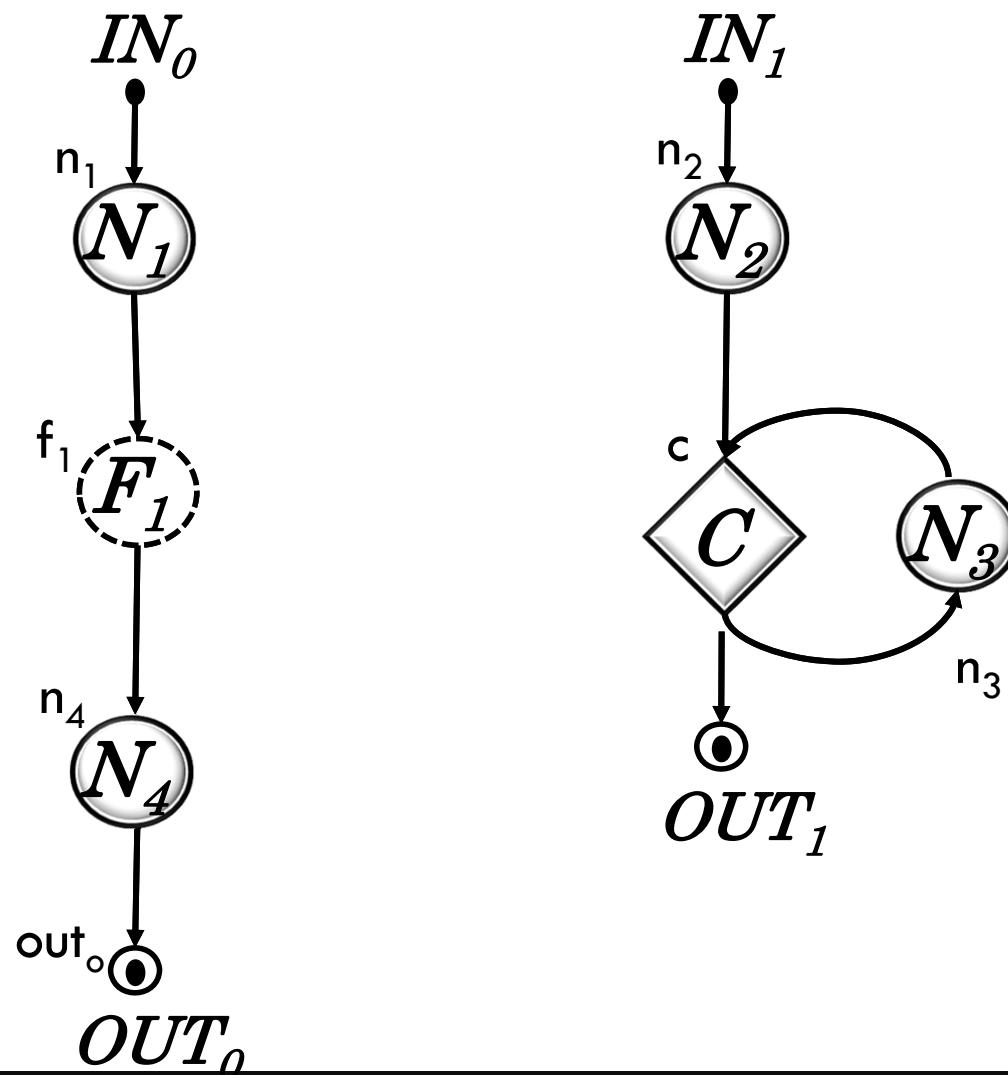


Instrumentation example



Structure & Timing information

Initialization: S

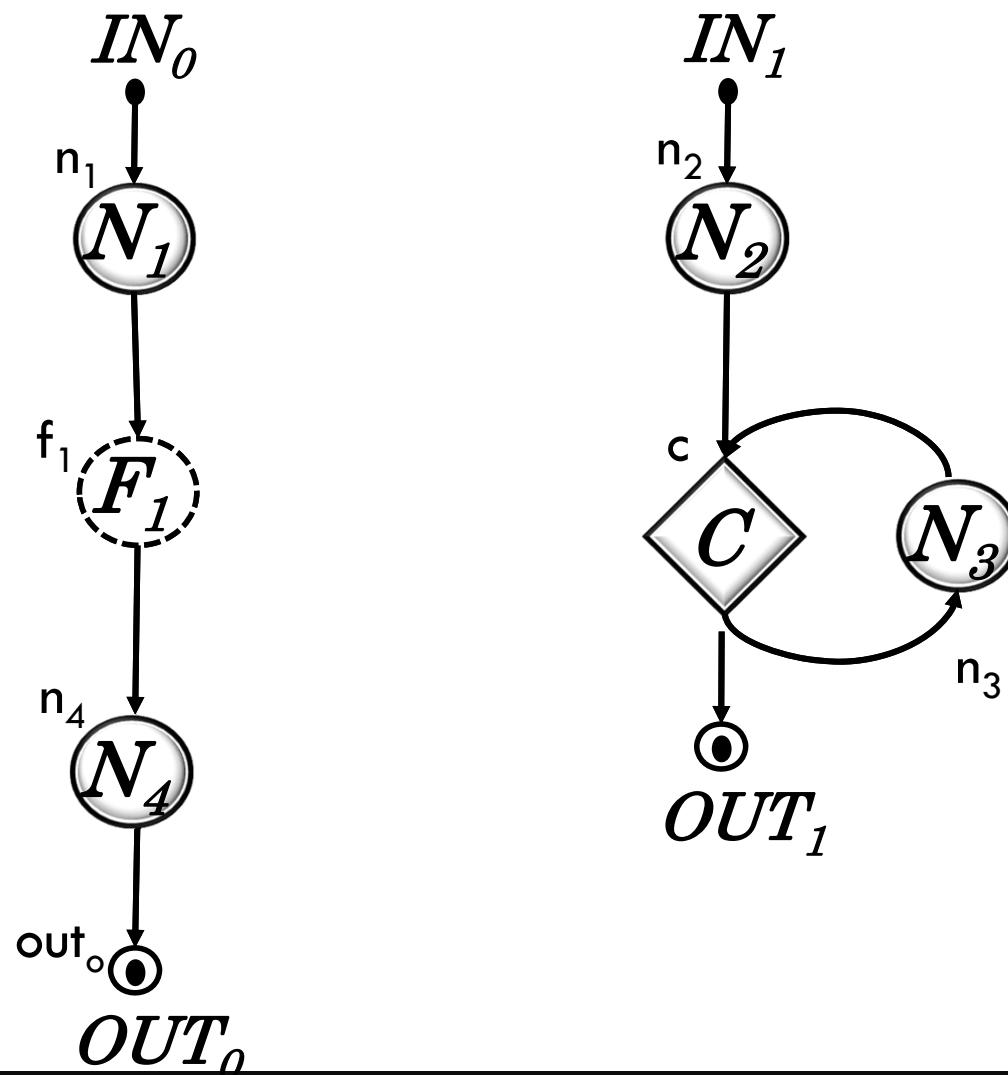


Structure & Timing information

Initialization:

S

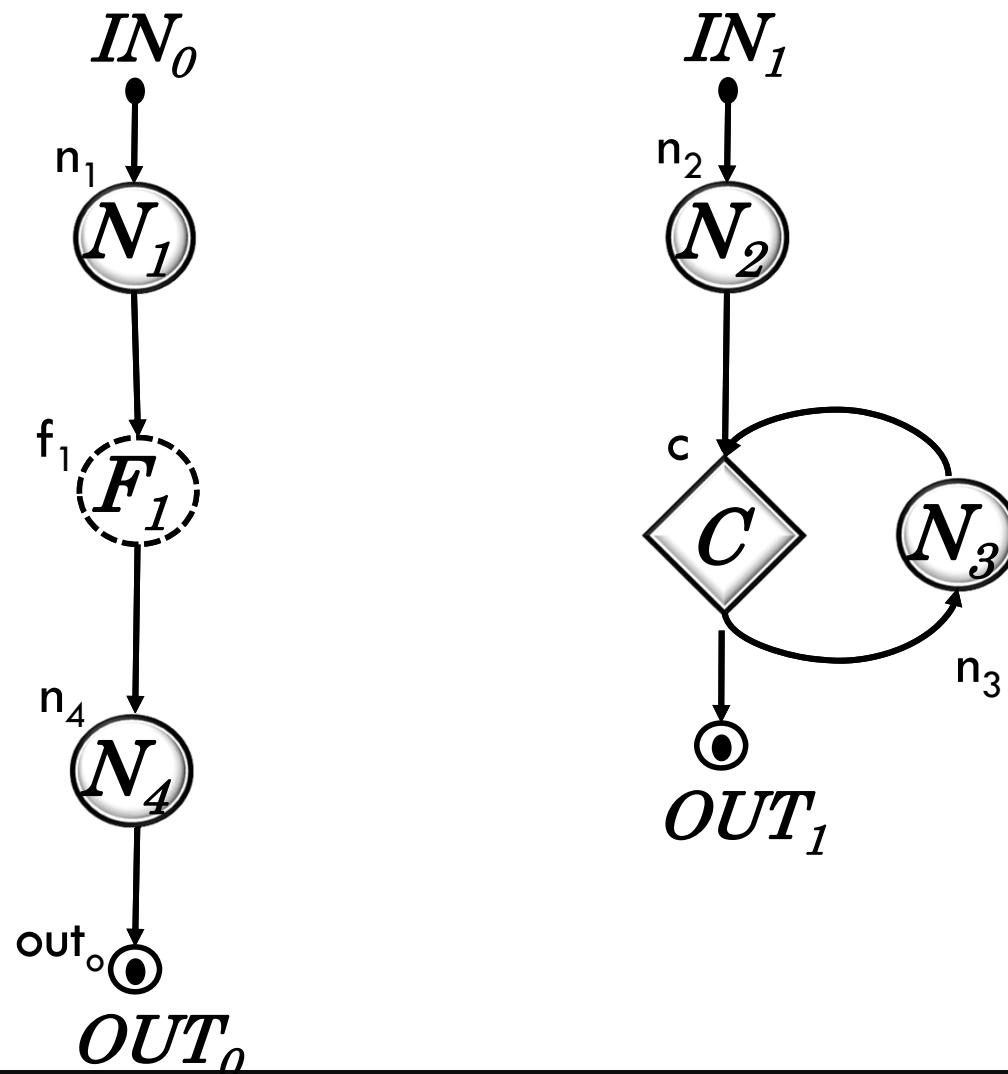
- Structure
 - Level



Structure & Timing information

Initialization:  S

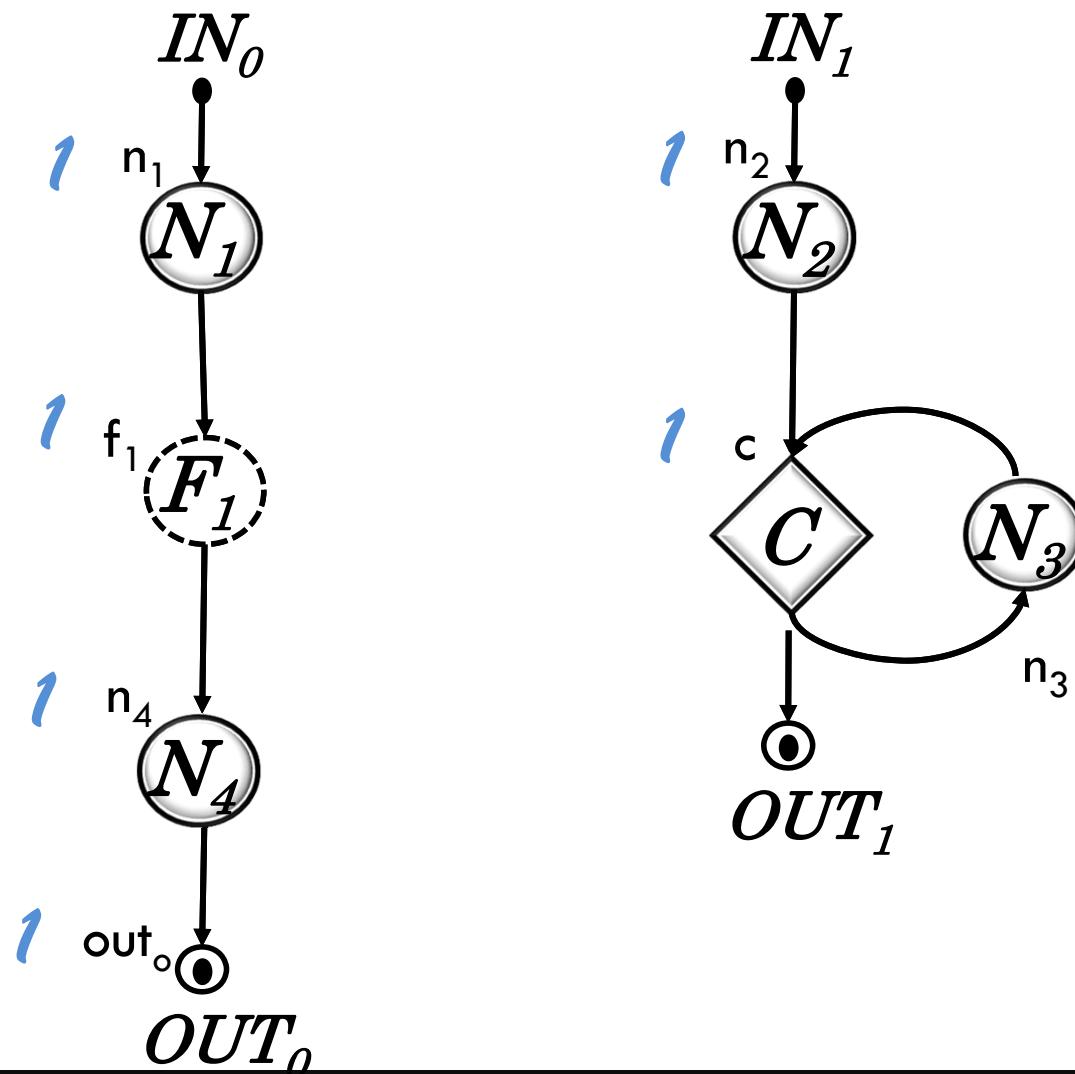
- Structure
 - Level



Structure & Timing information

Initialization: \emptyset S

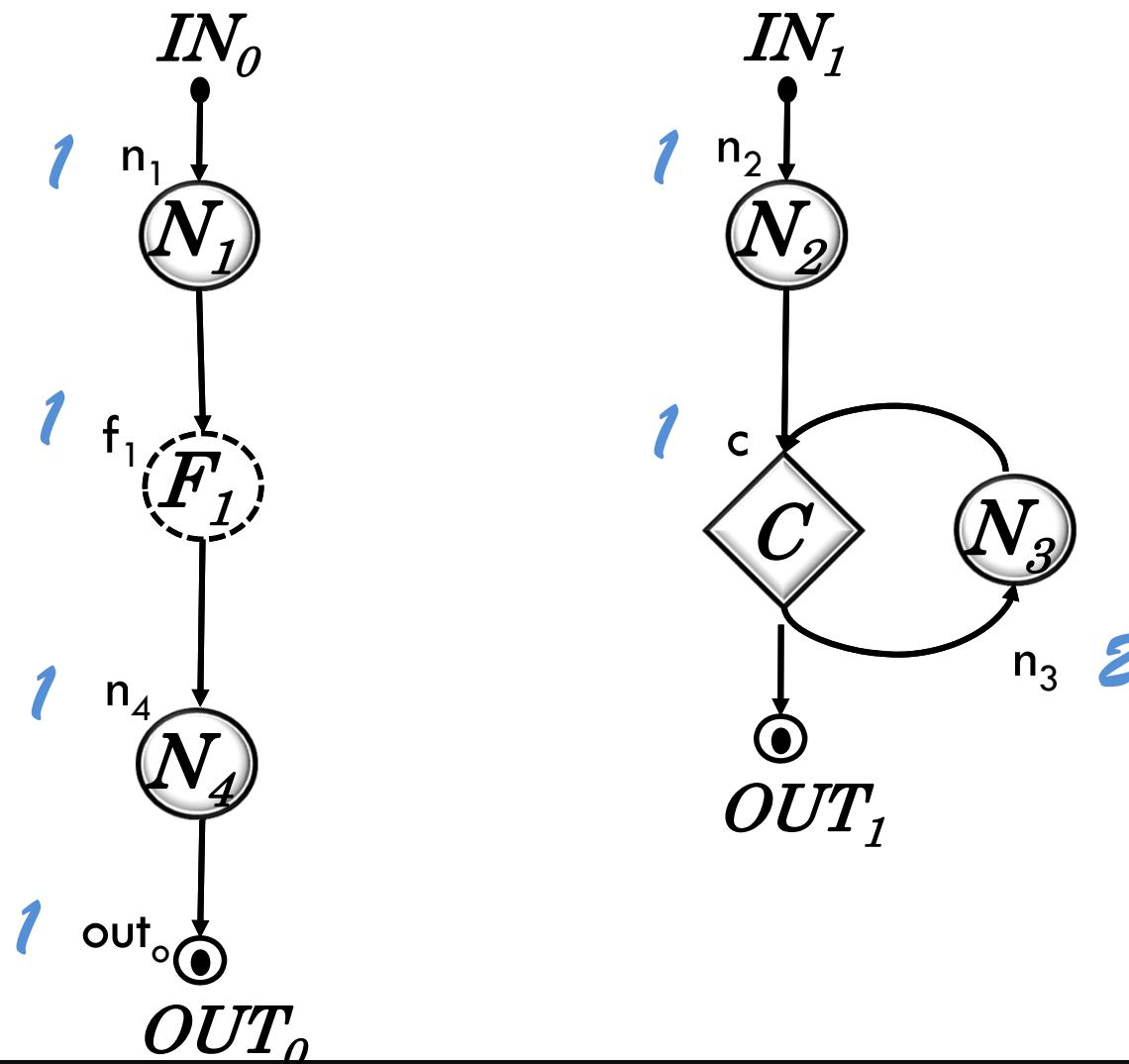
- Structure
 - Level



Structure & Timing information

Initialization: \emptyset S

- Structure
 - Level

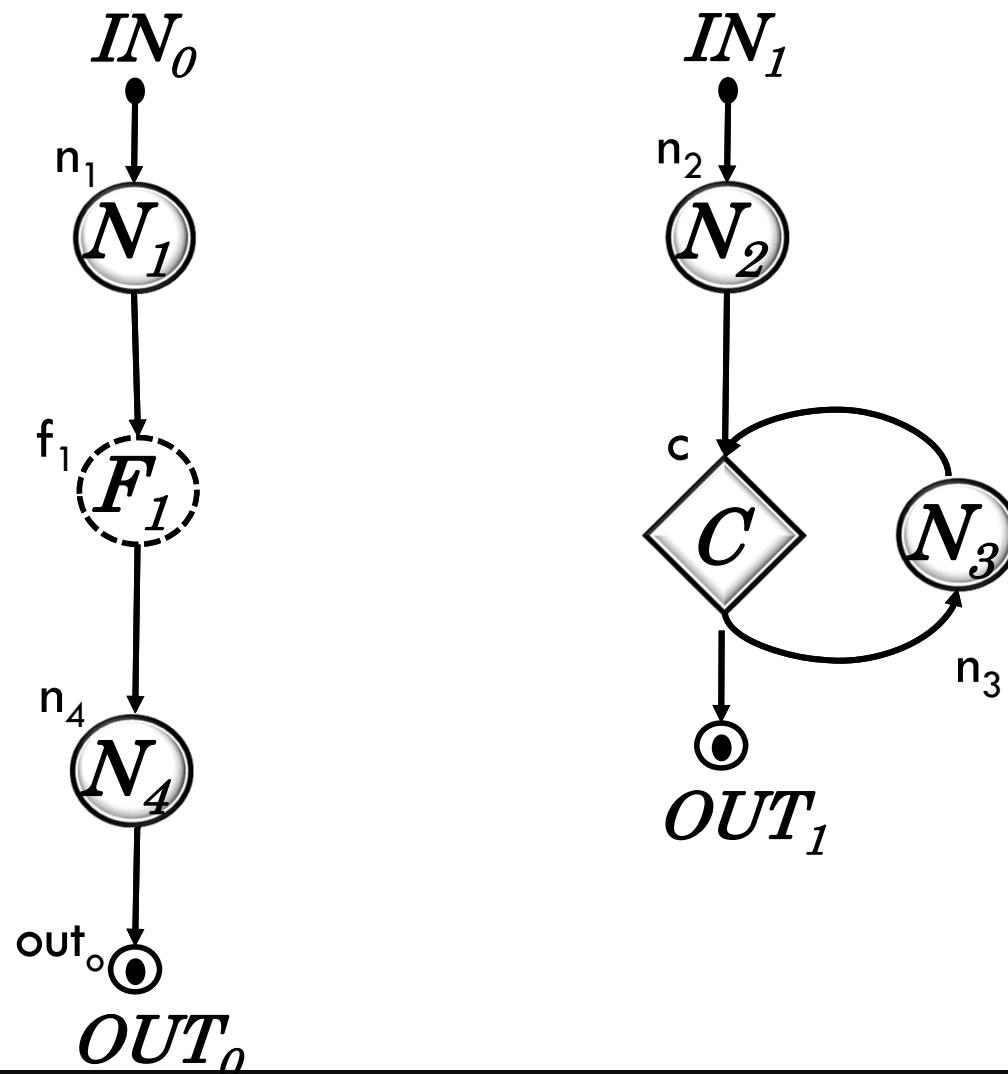


Structure & Timing information

Initialization:

S

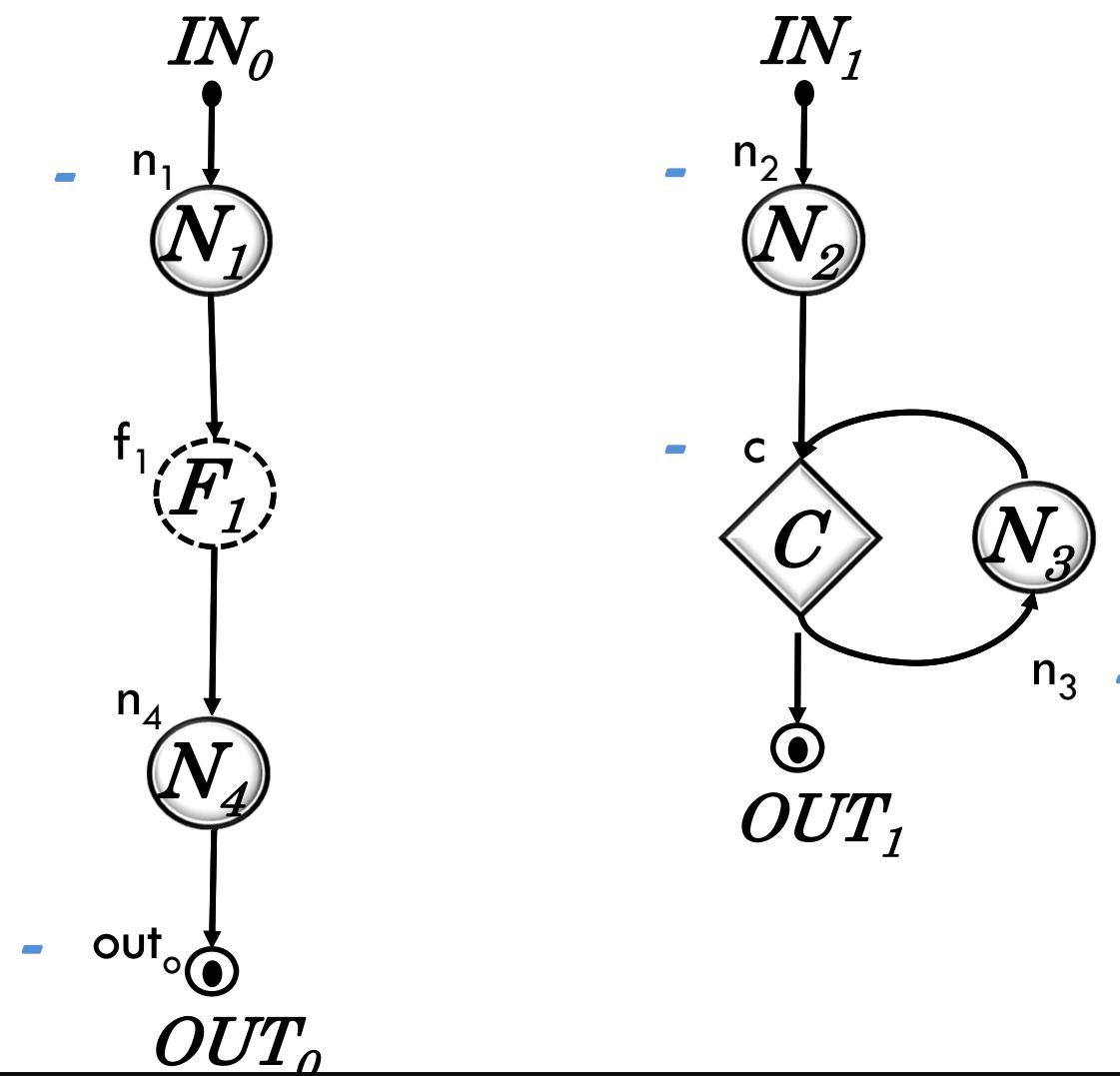
- Structure
 - Level
 - Type



Structure & Timing information

Initialization: - S

- Structure
 - Level
 - Type



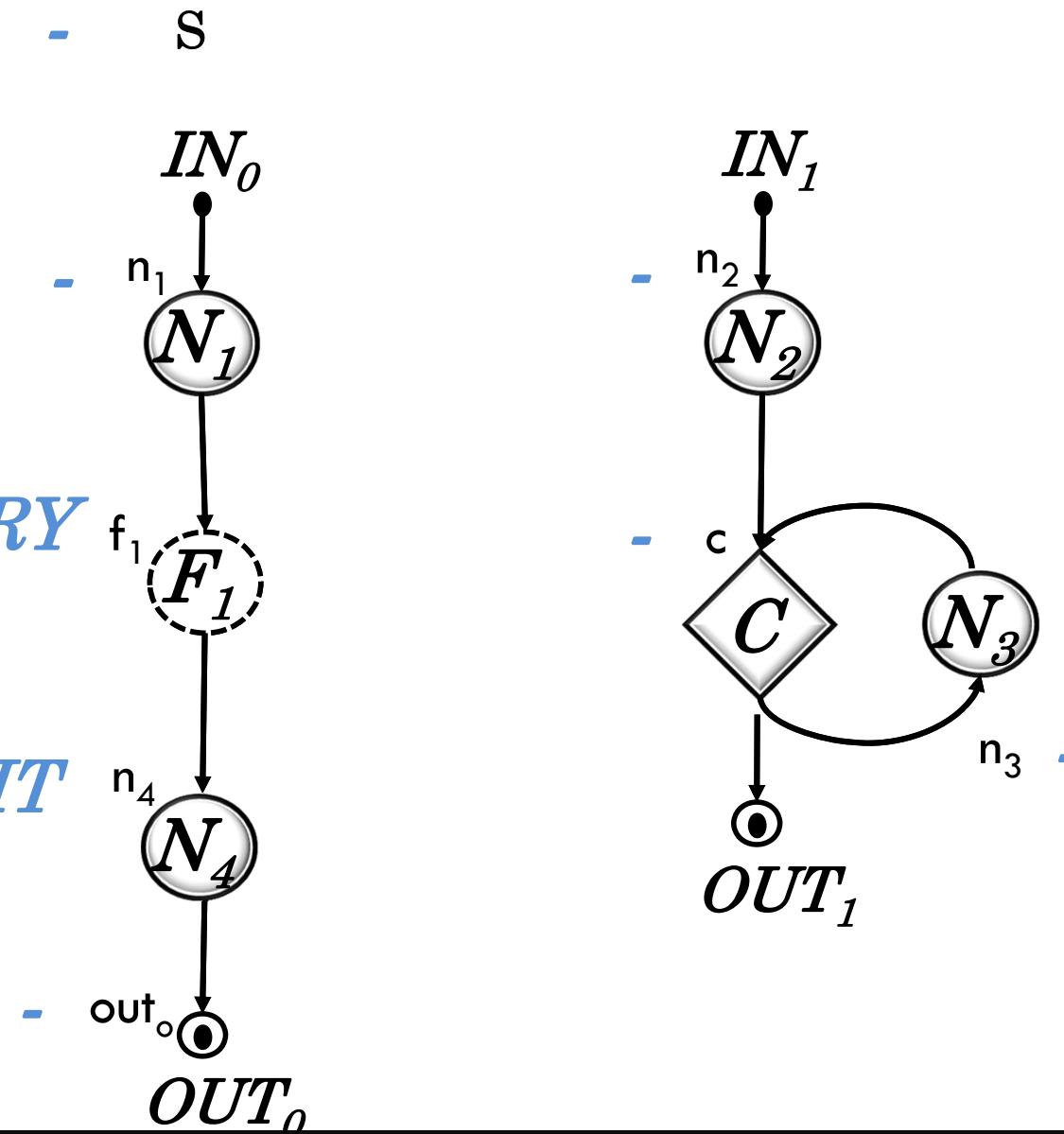
Structure & Timing information

Initialization: - S

- Structure
 - Level
 - Type

F_ENTRY

F_EXIT

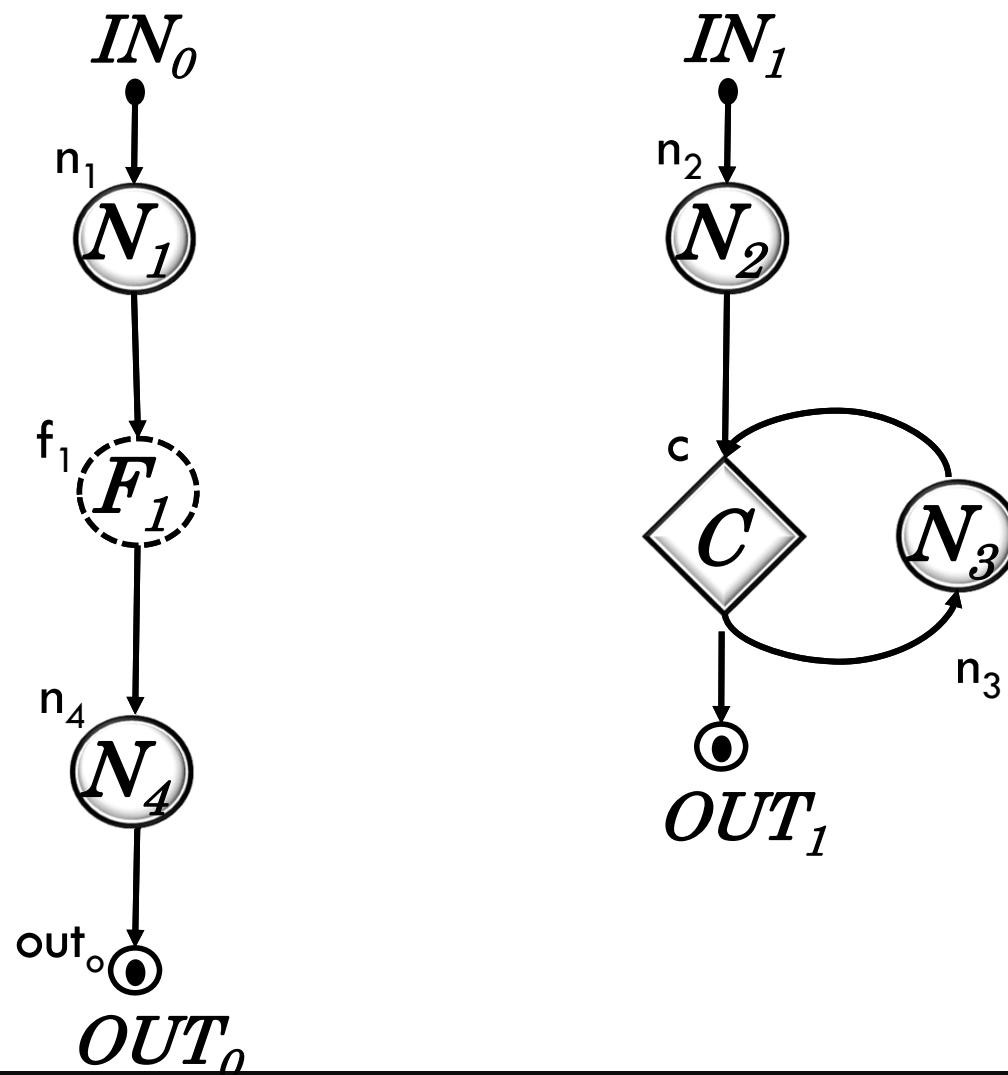


Structure & Timing information

Initialization:

S

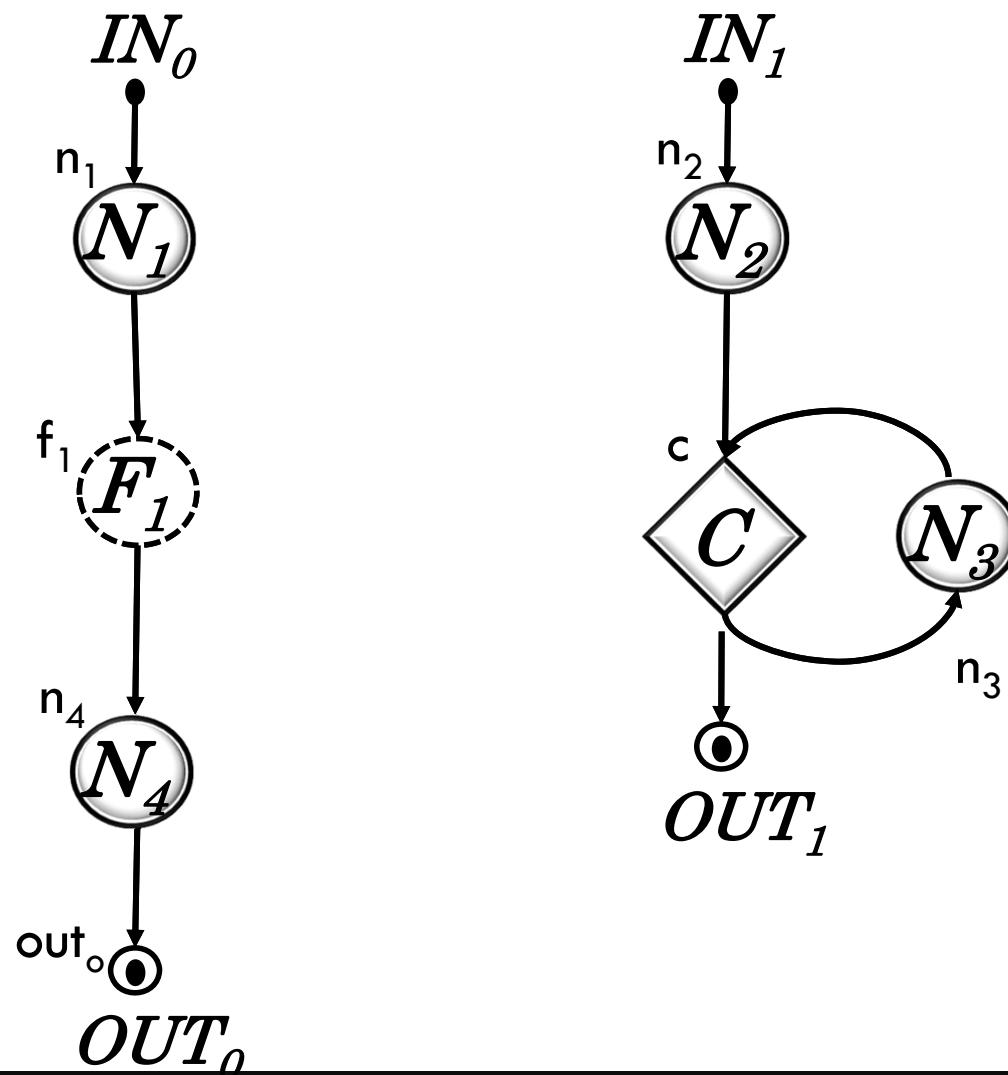
- Structure
 - Level
 - Type
 - Head



Structure & Timing information

Initialization: - S

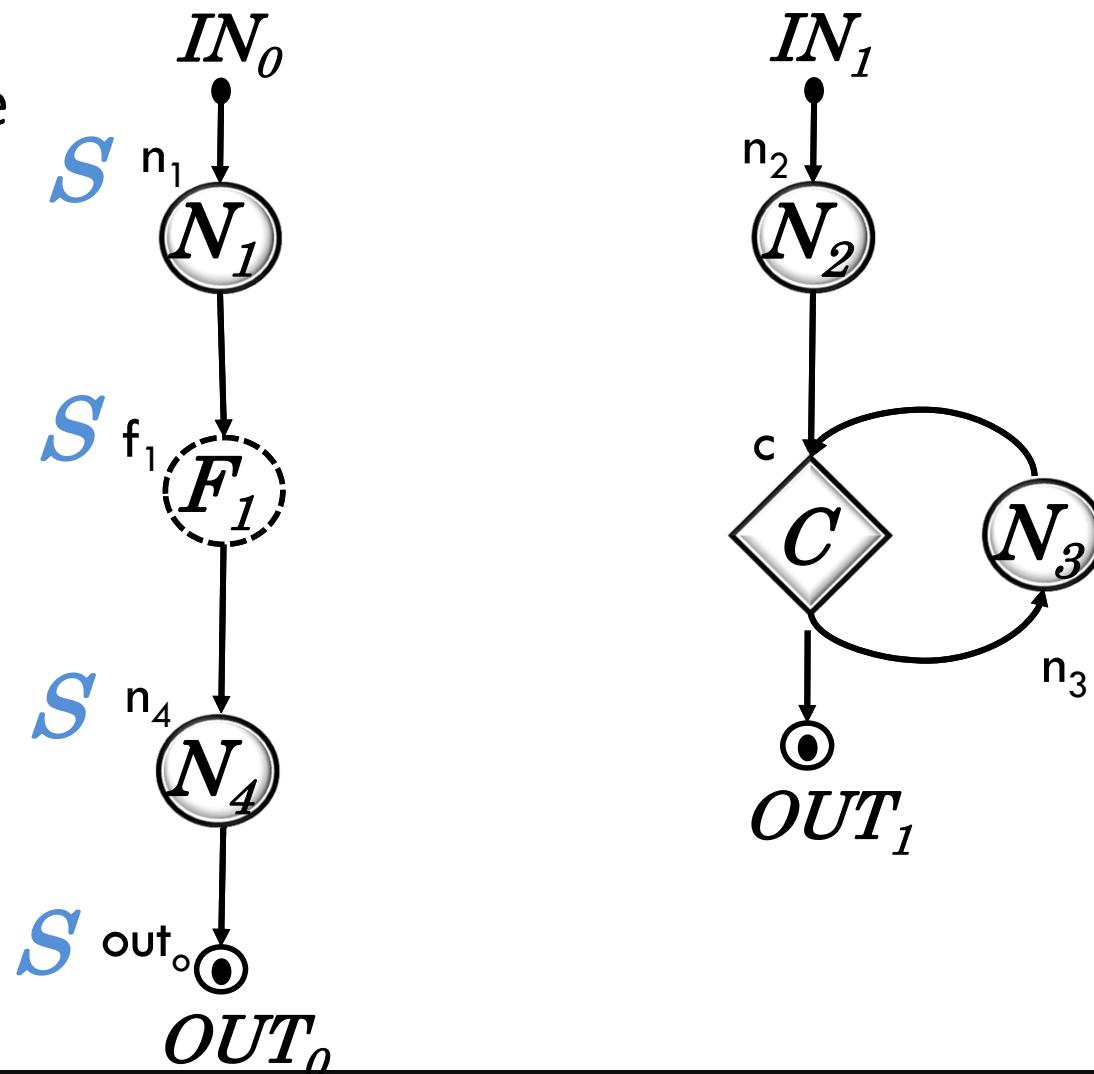
- Structure
 - Level
 - Type
 - Head



Structure & Timing information

Initialization: - S

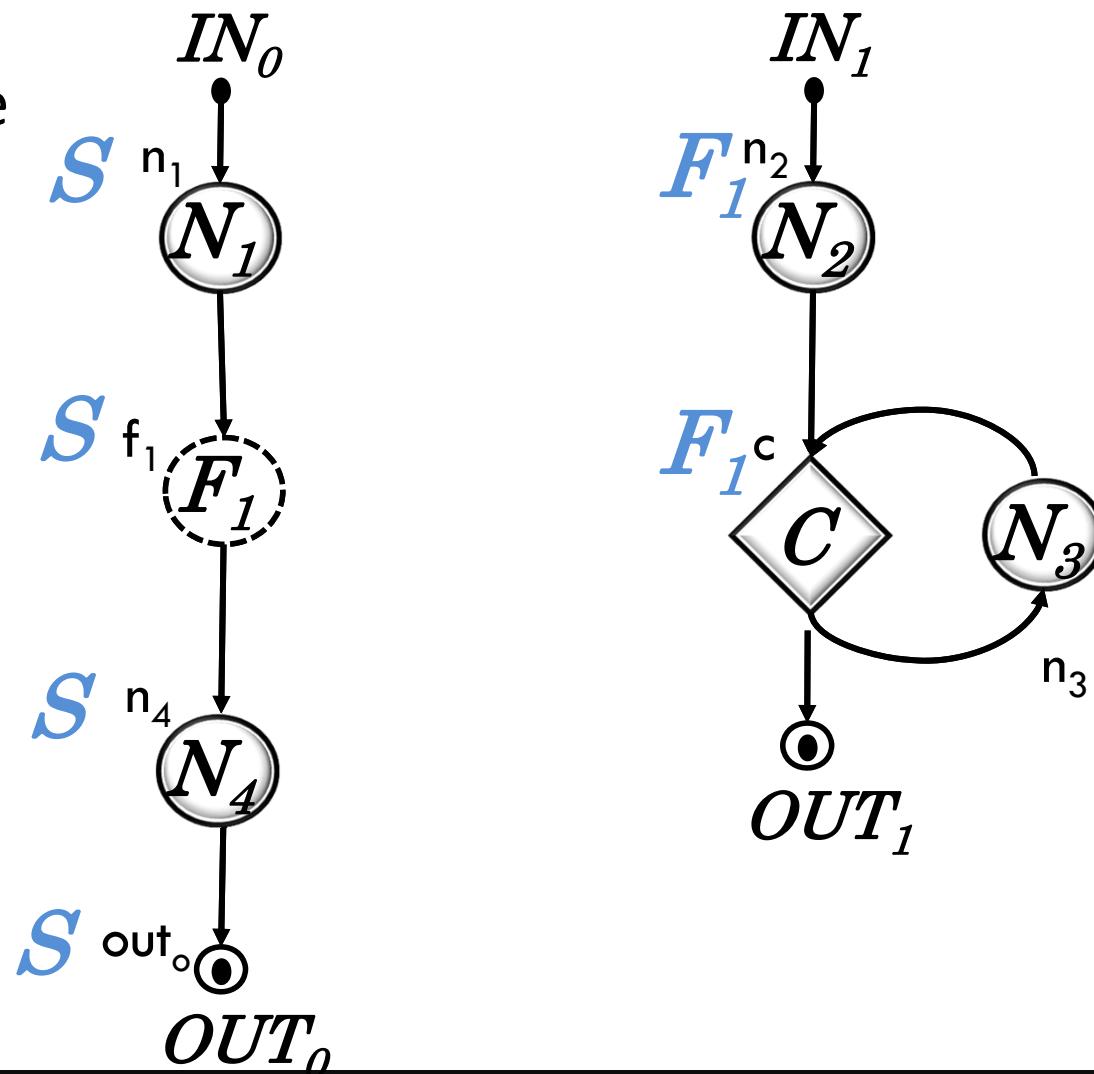
- Structure
 - Level
 - Type
 - Head



Structure & Timing information

Initialization: - S

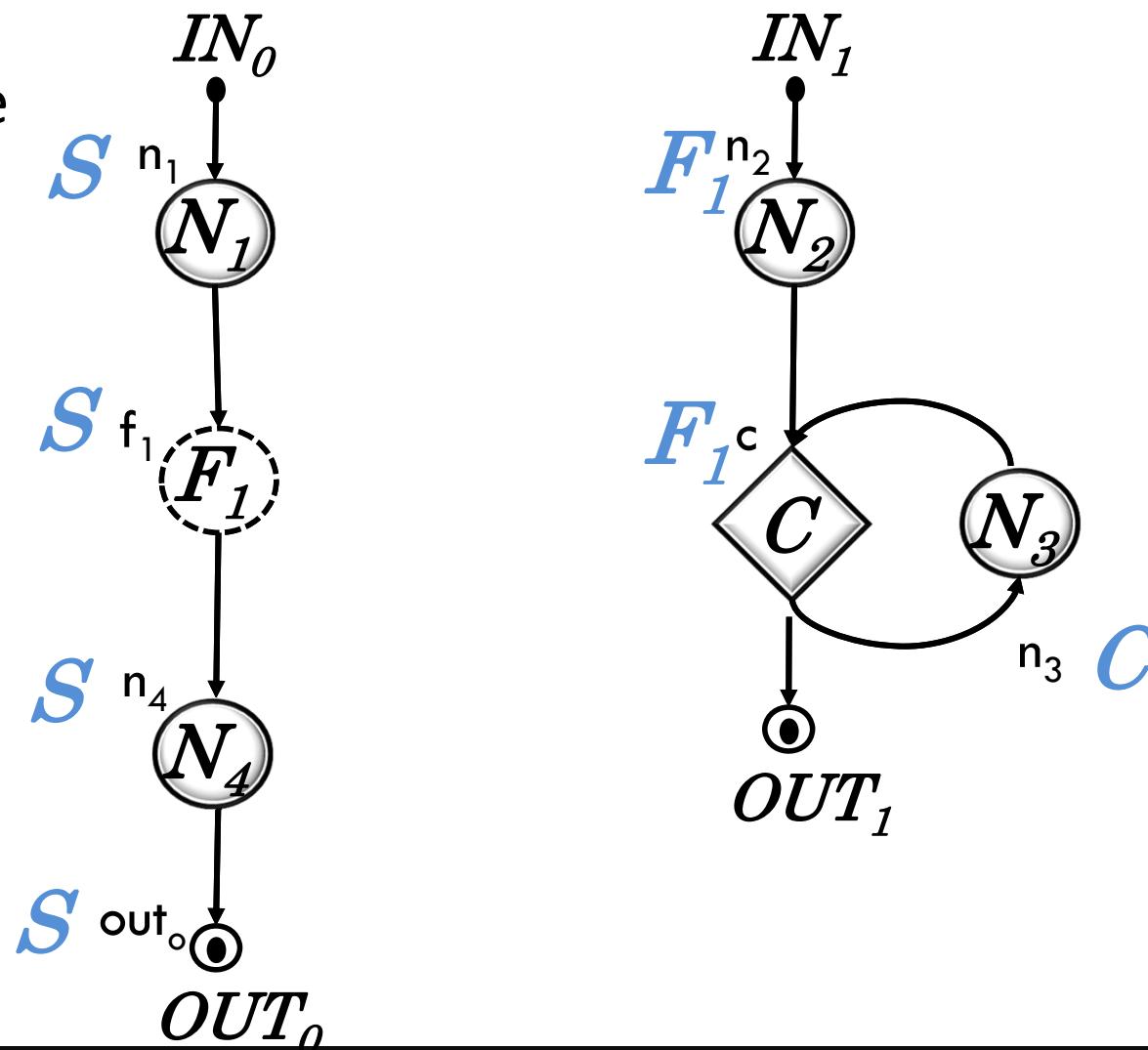
- Structure
 - Level
 - Type
 - Head



Structure & Timing information

Initialization: - S

- Structure
 - Level
 - Type
 - Head

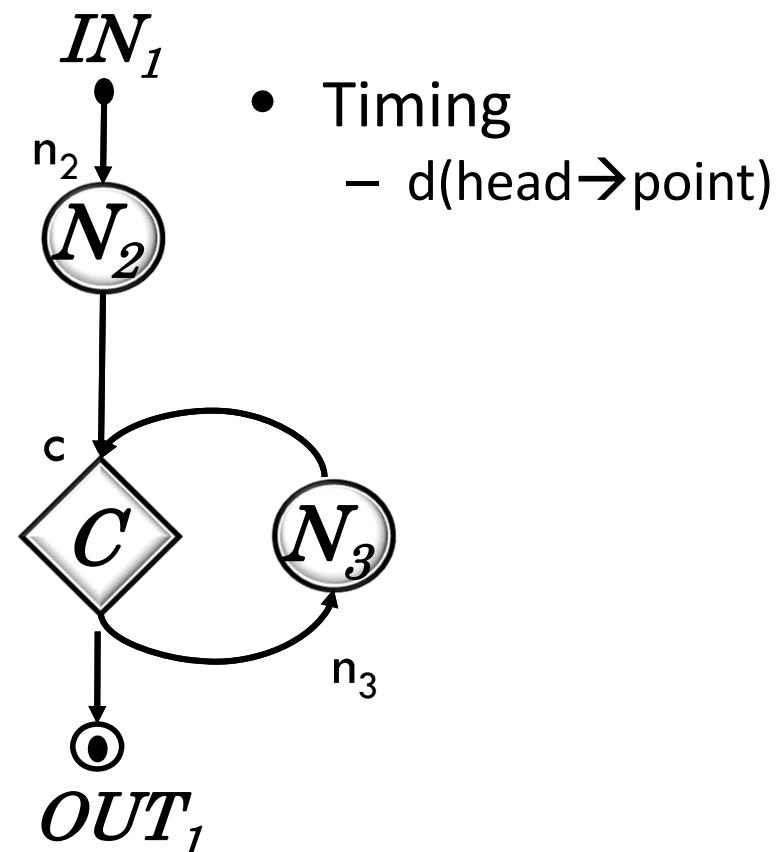
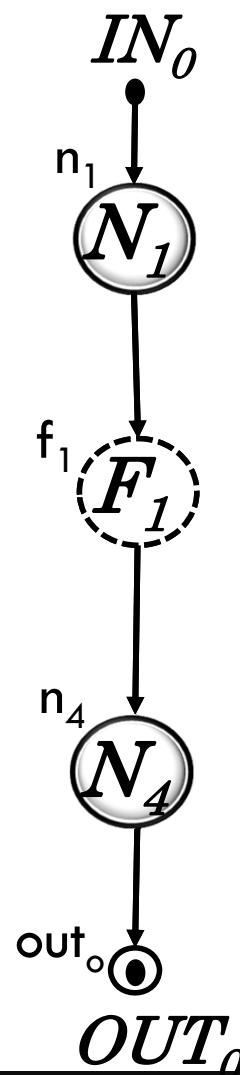


Structure & Timing information

Initialization:

S

- Structure
 - Level
 - Type
 - Head

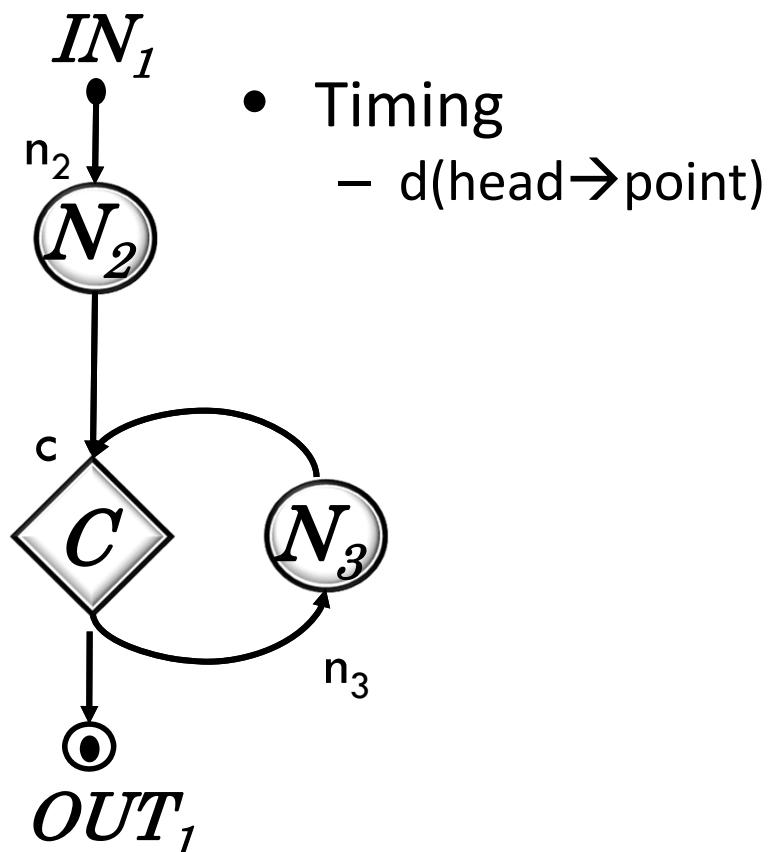
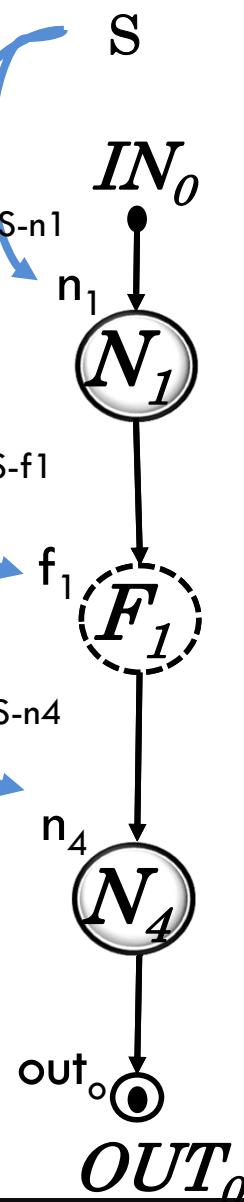


- Timing
 - $d(\text{head} \rightarrow \text{point})$

Structure & Timing information

Initialization:

- Structure
 - Level
 - Type
 - Head



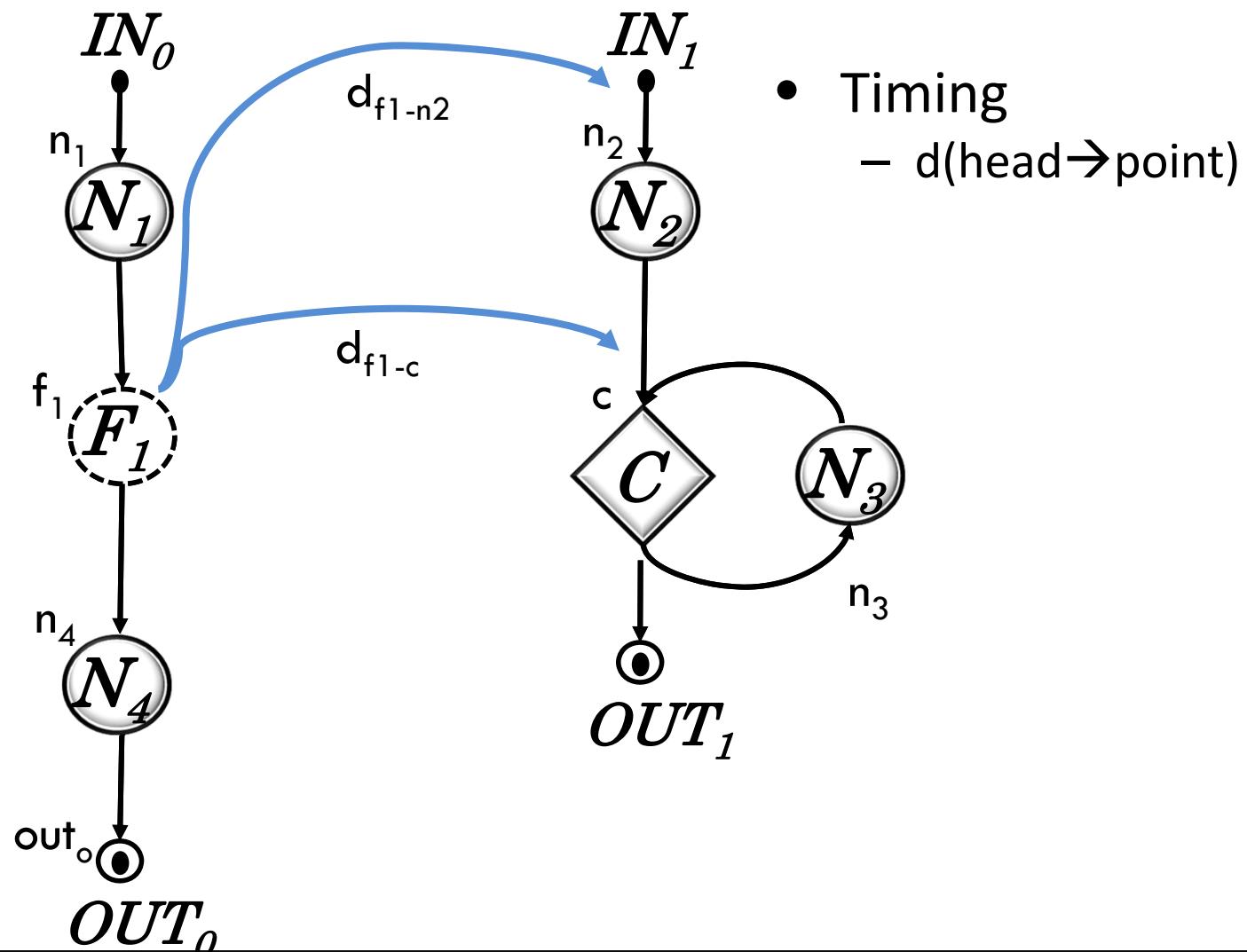
- Timing
 - $d(\text{head} \rightarrow \text{point})$

Structure & Timing information

Initialization:

- Structure
 - Level
 - Type
 - Head

S

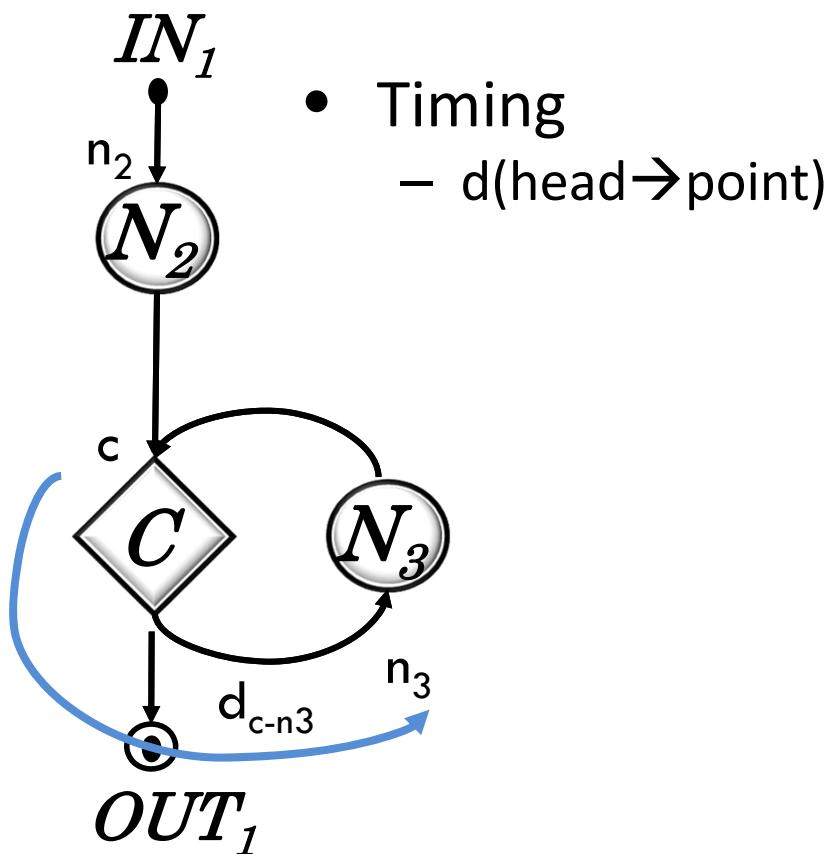
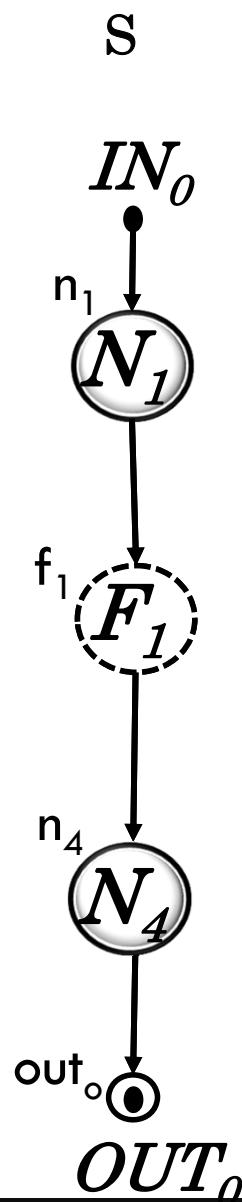


- Timing
 - $d(\text{head} \rightarrow \text{point})$

Structure & Timing information

Initialization:

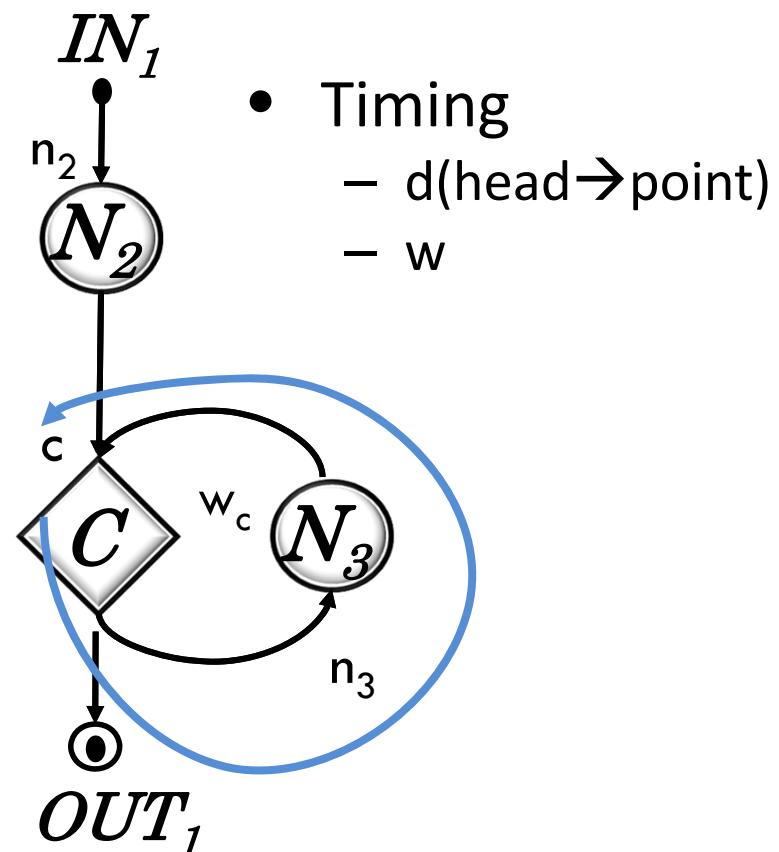
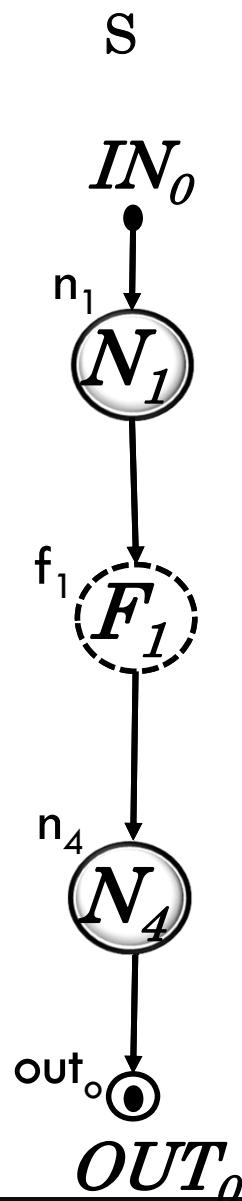
- Structure
 - Level
 - Type
 - Head



Structure & Timing information

Initialization:

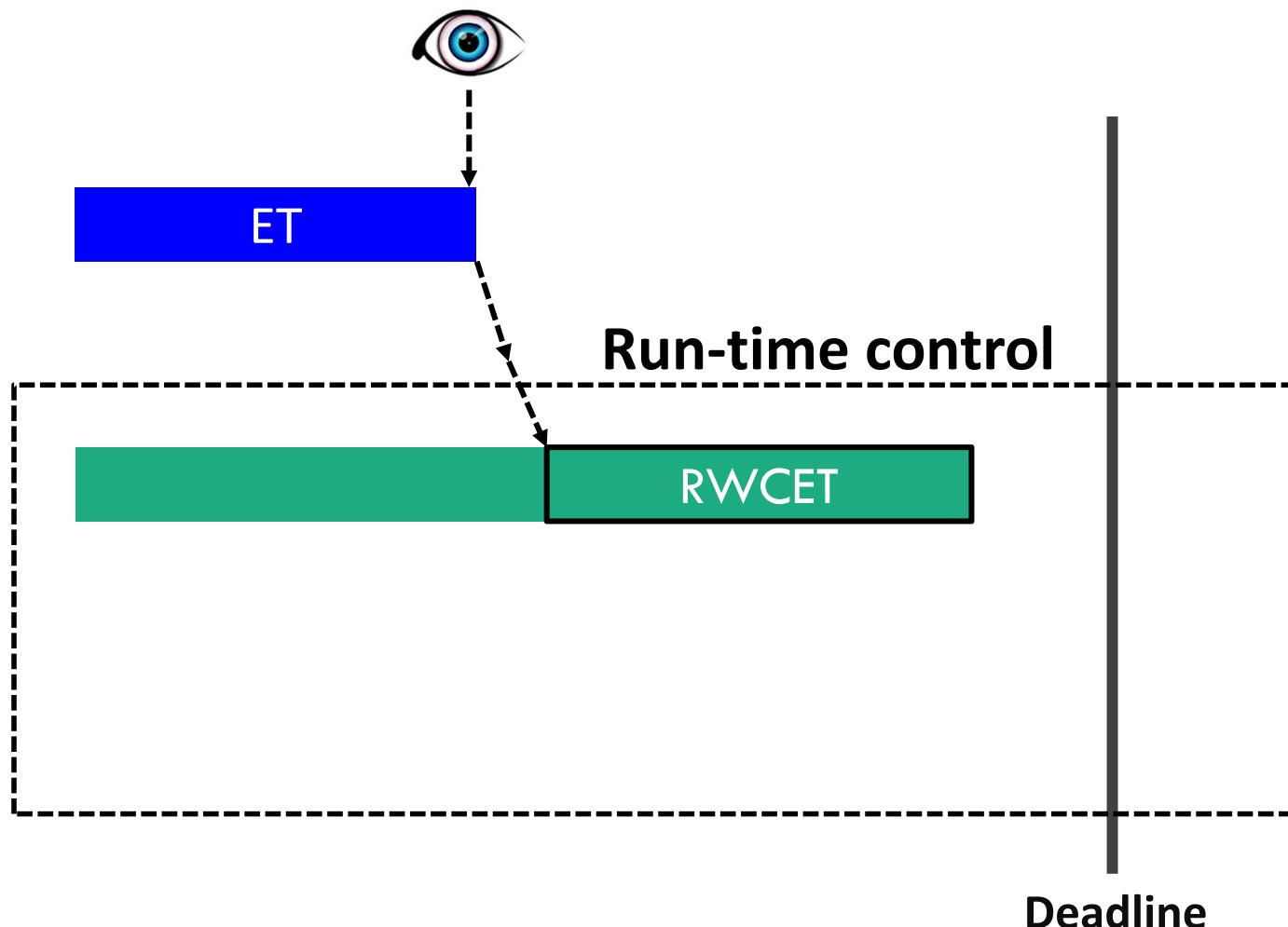
- Structure
 - Level
 - Type
 - Head



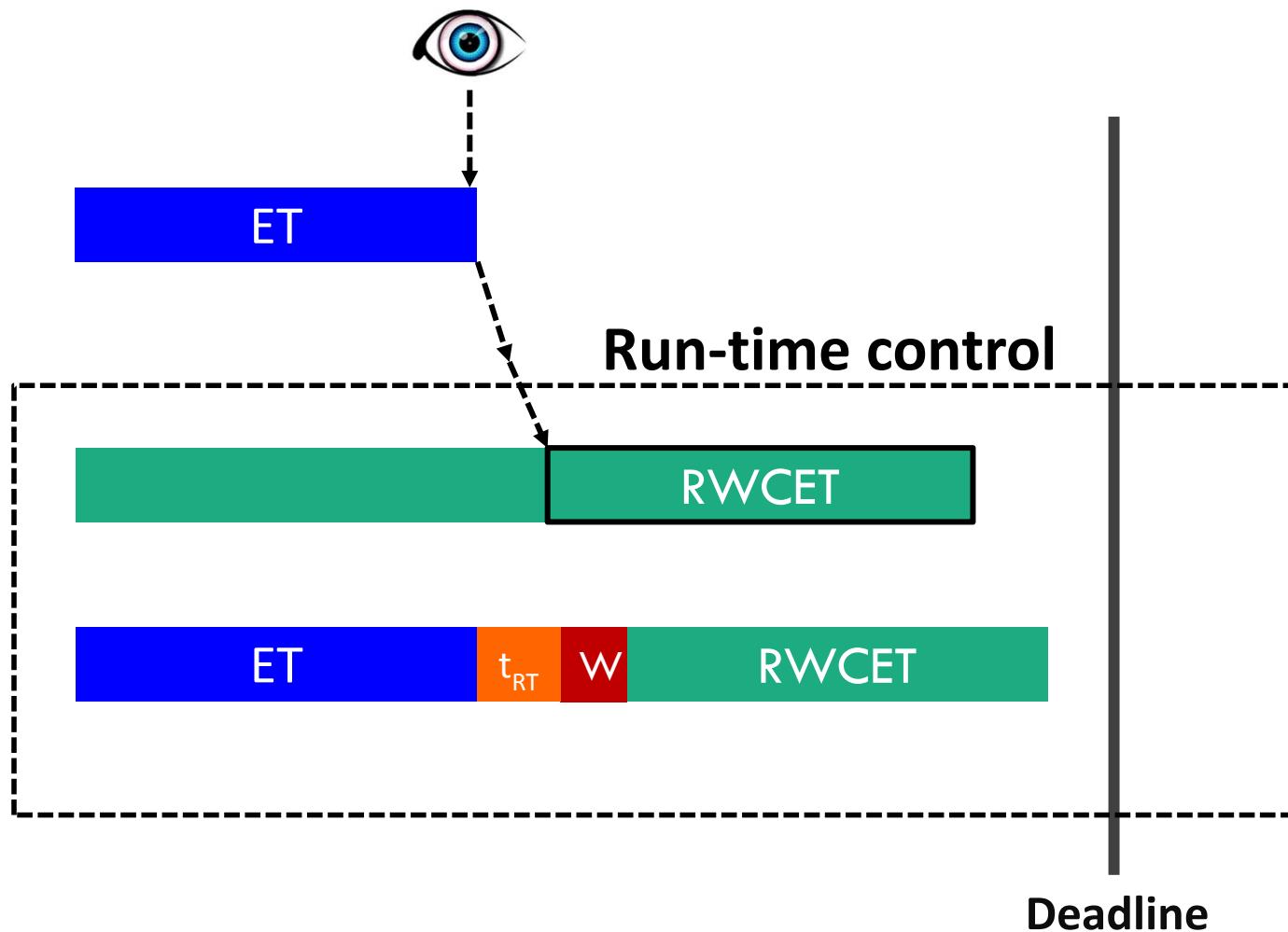
Observation point



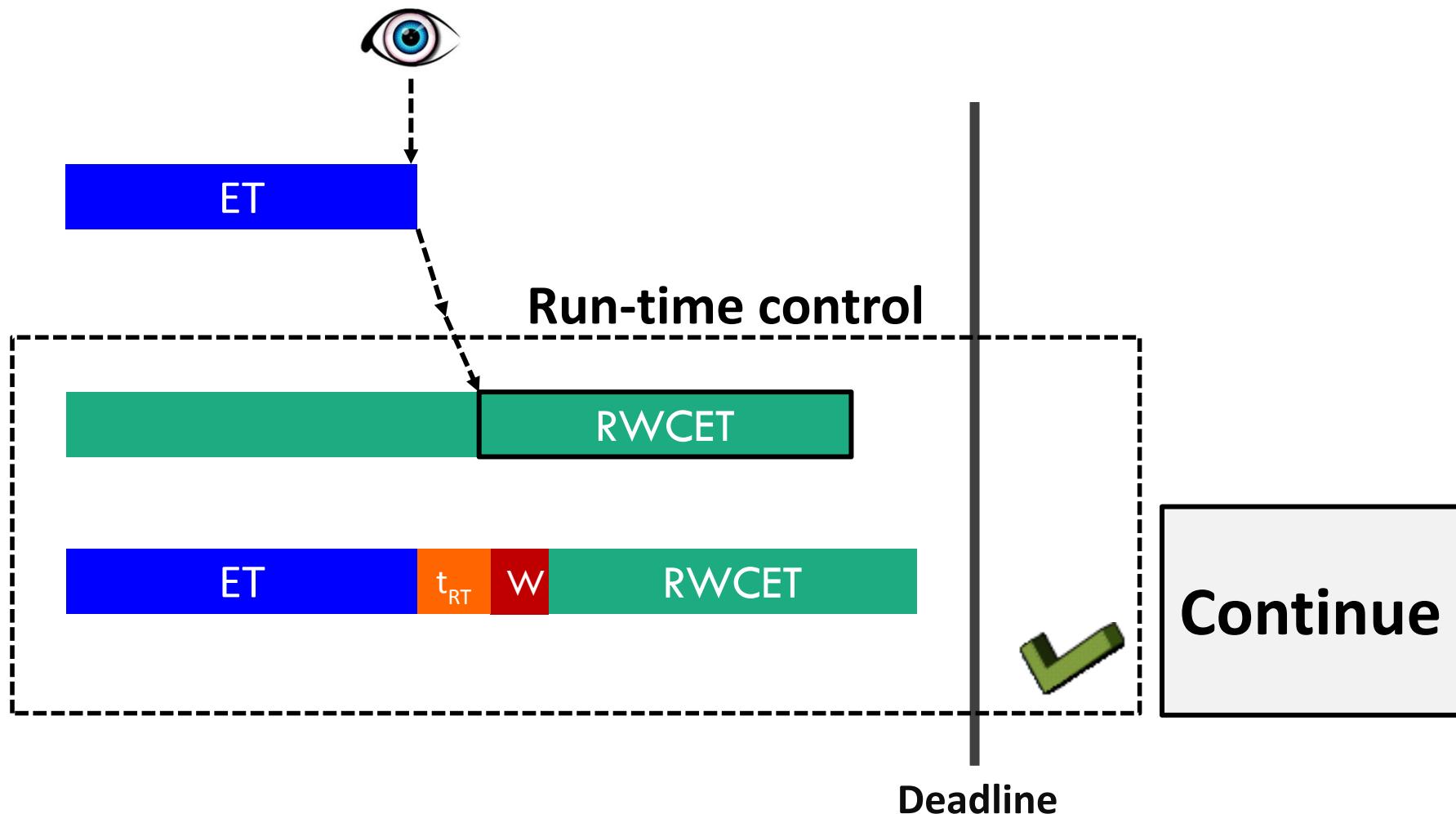
Observation point



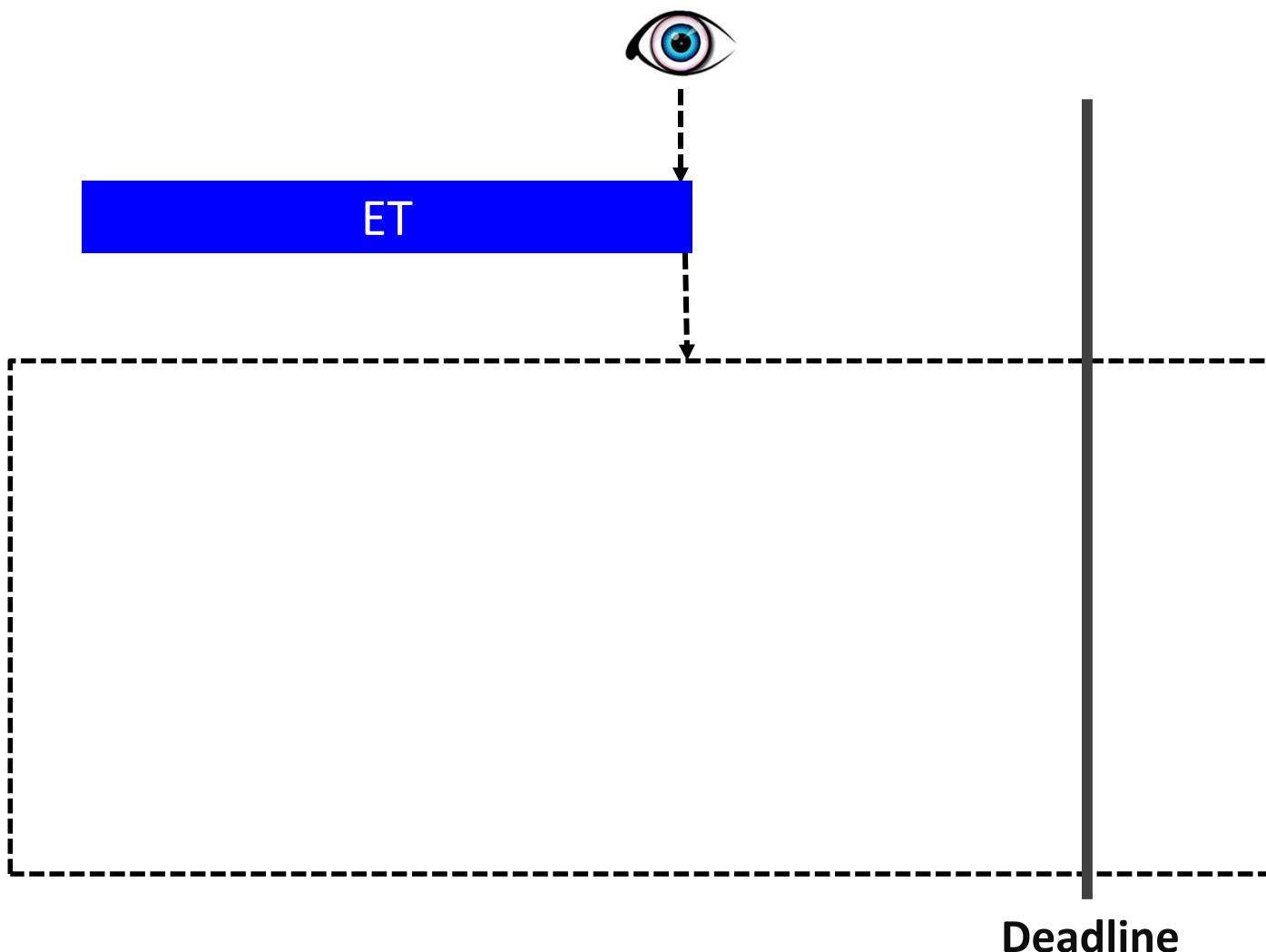
Observation point



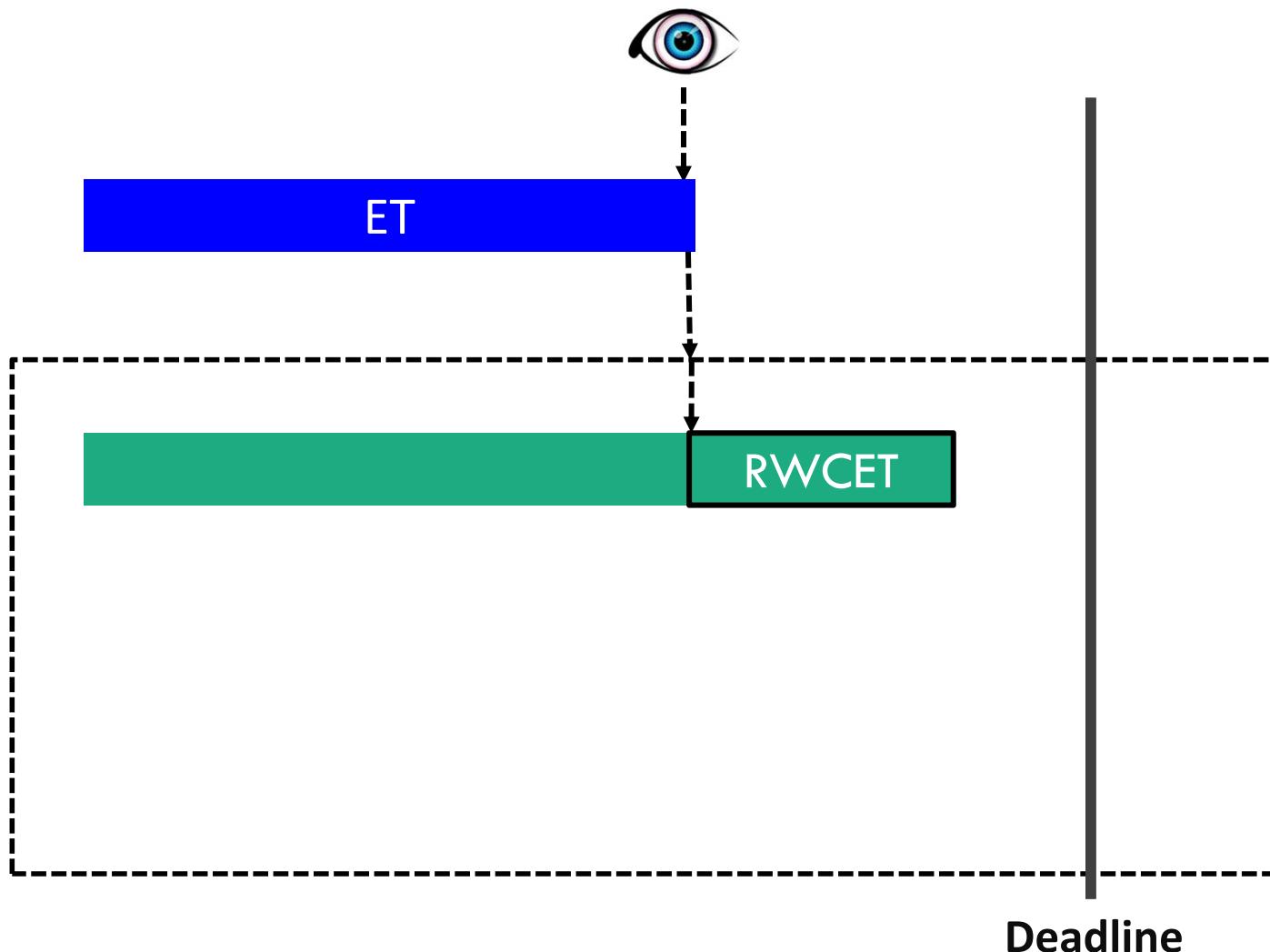
Observation point



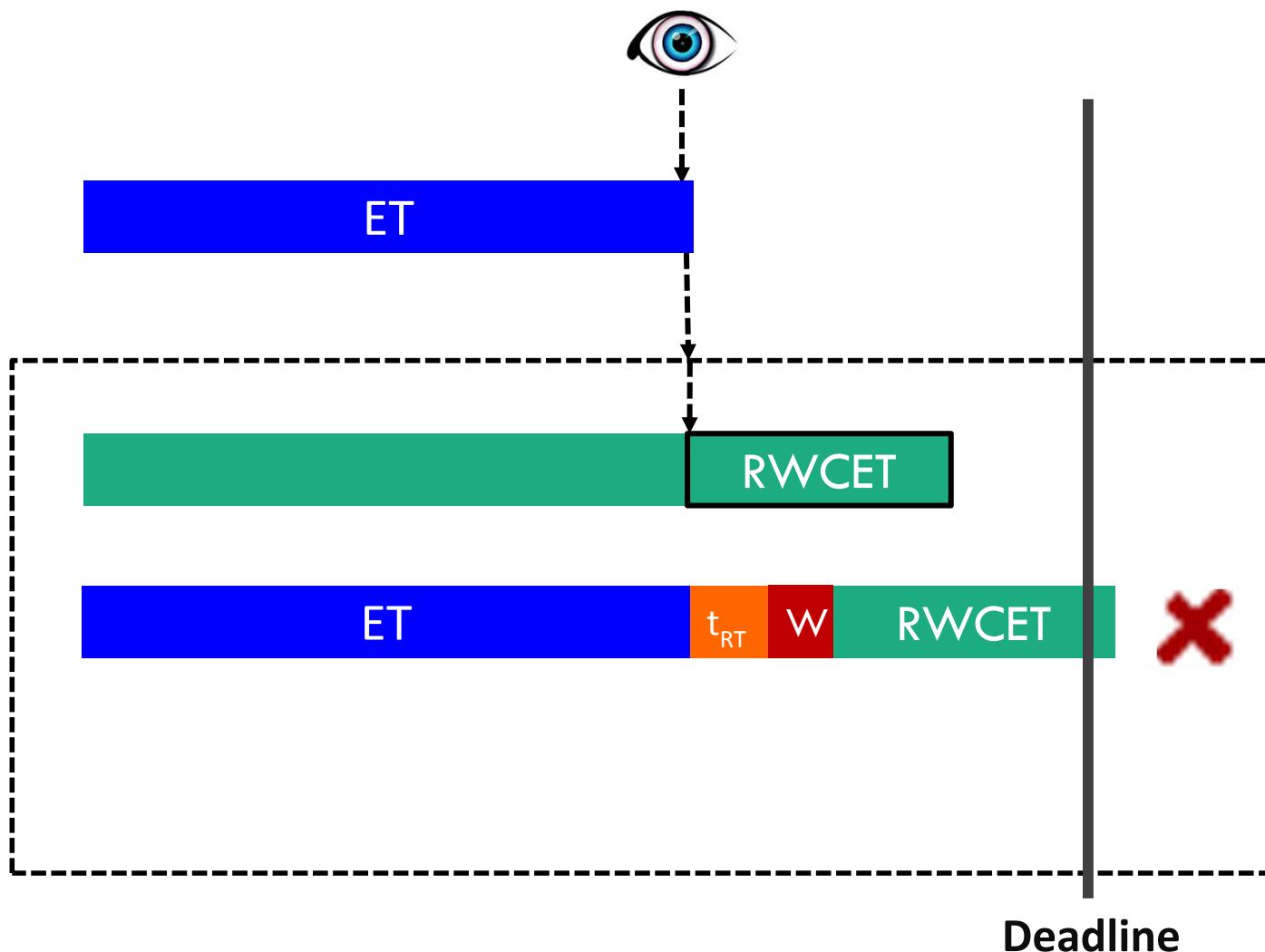
... later observation point



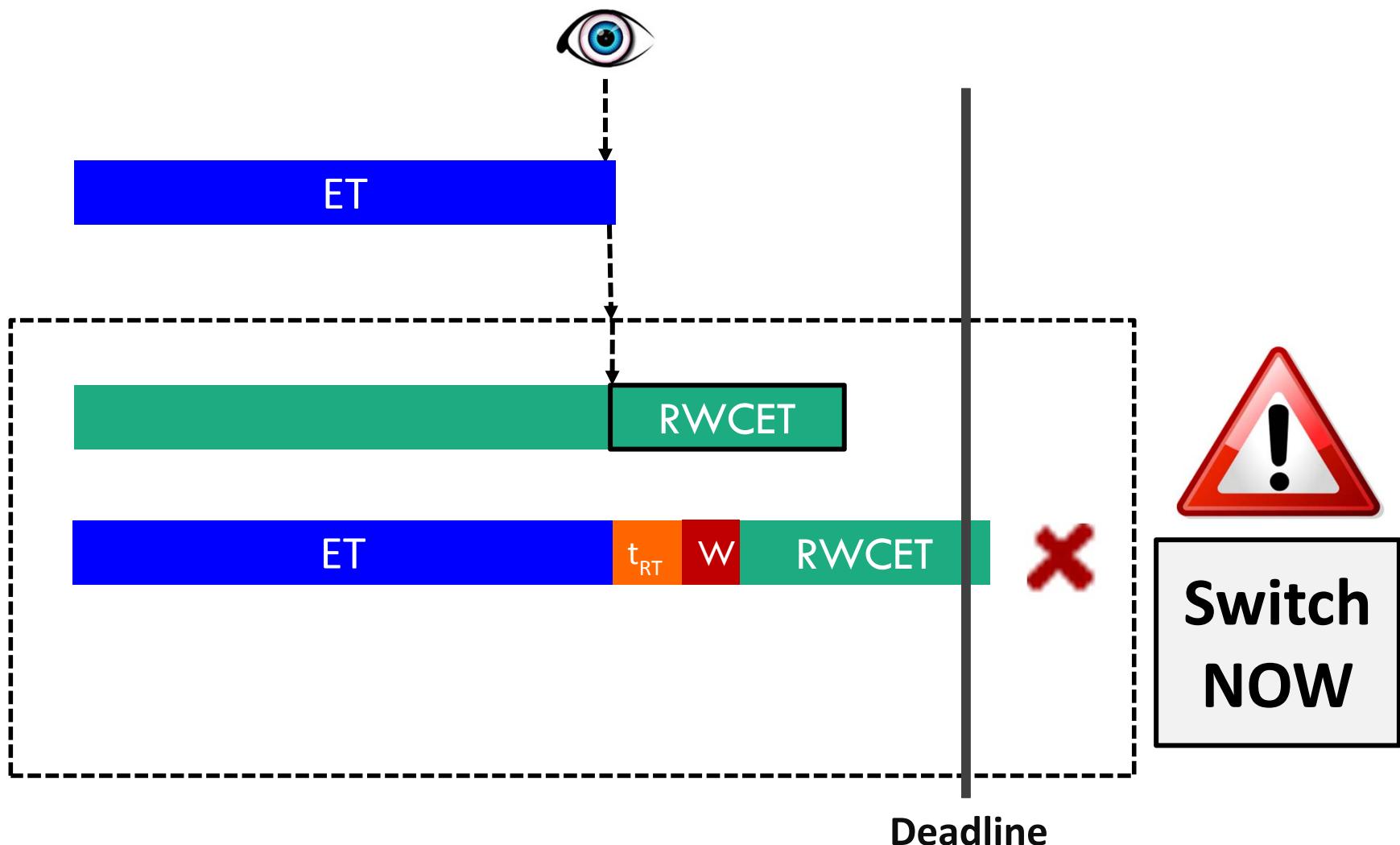
... later observation point



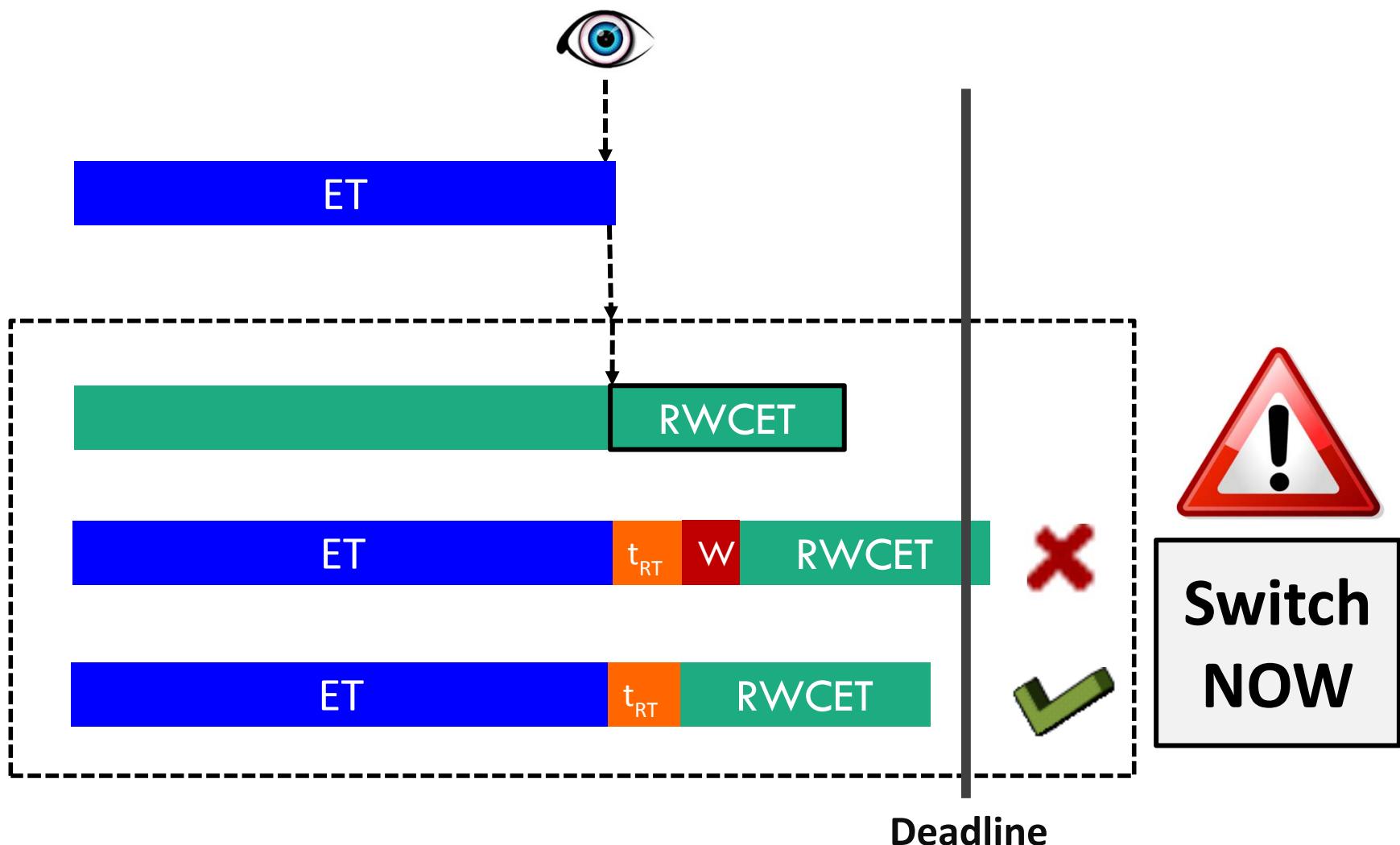
... later observation point



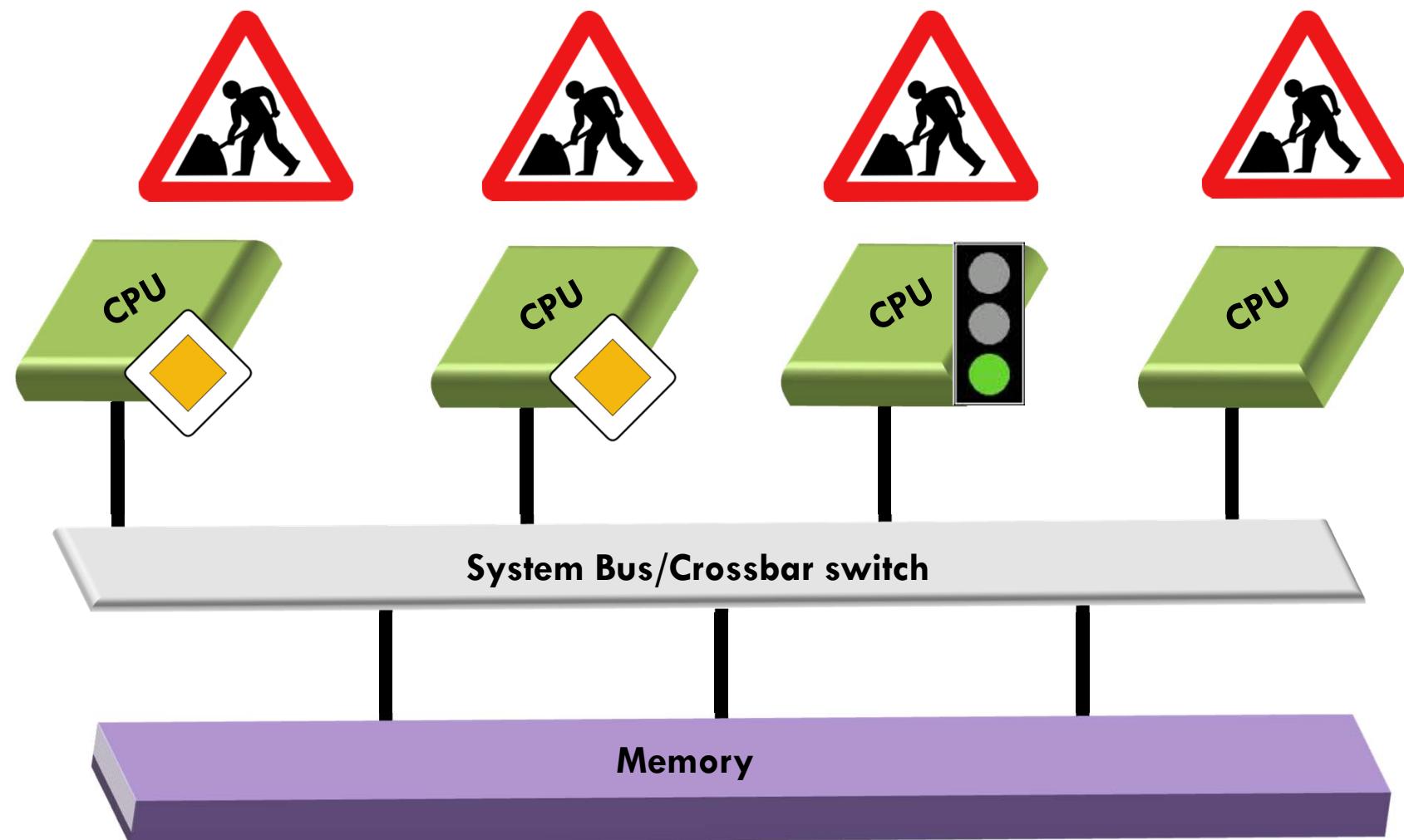
... later observation point



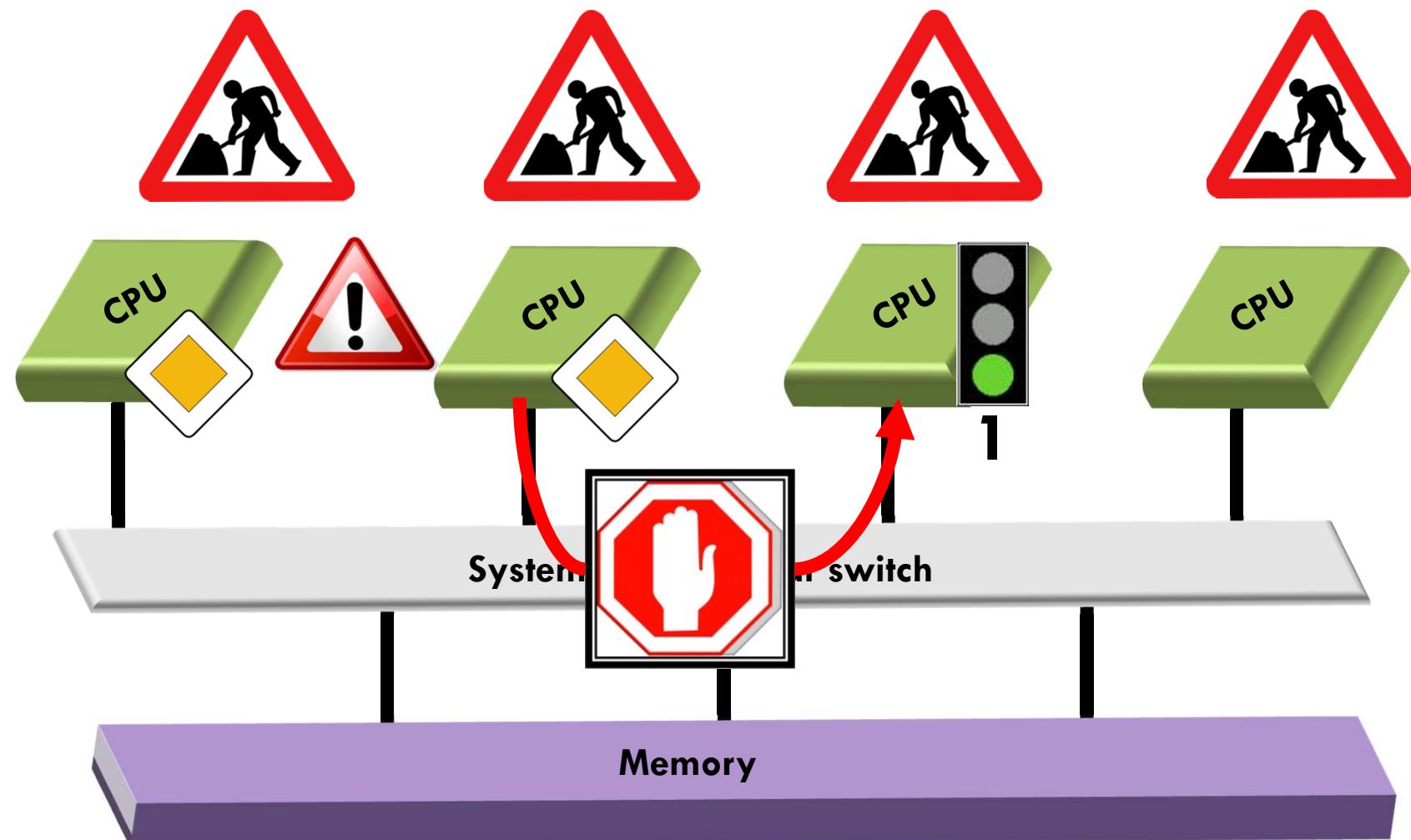
... later observation point



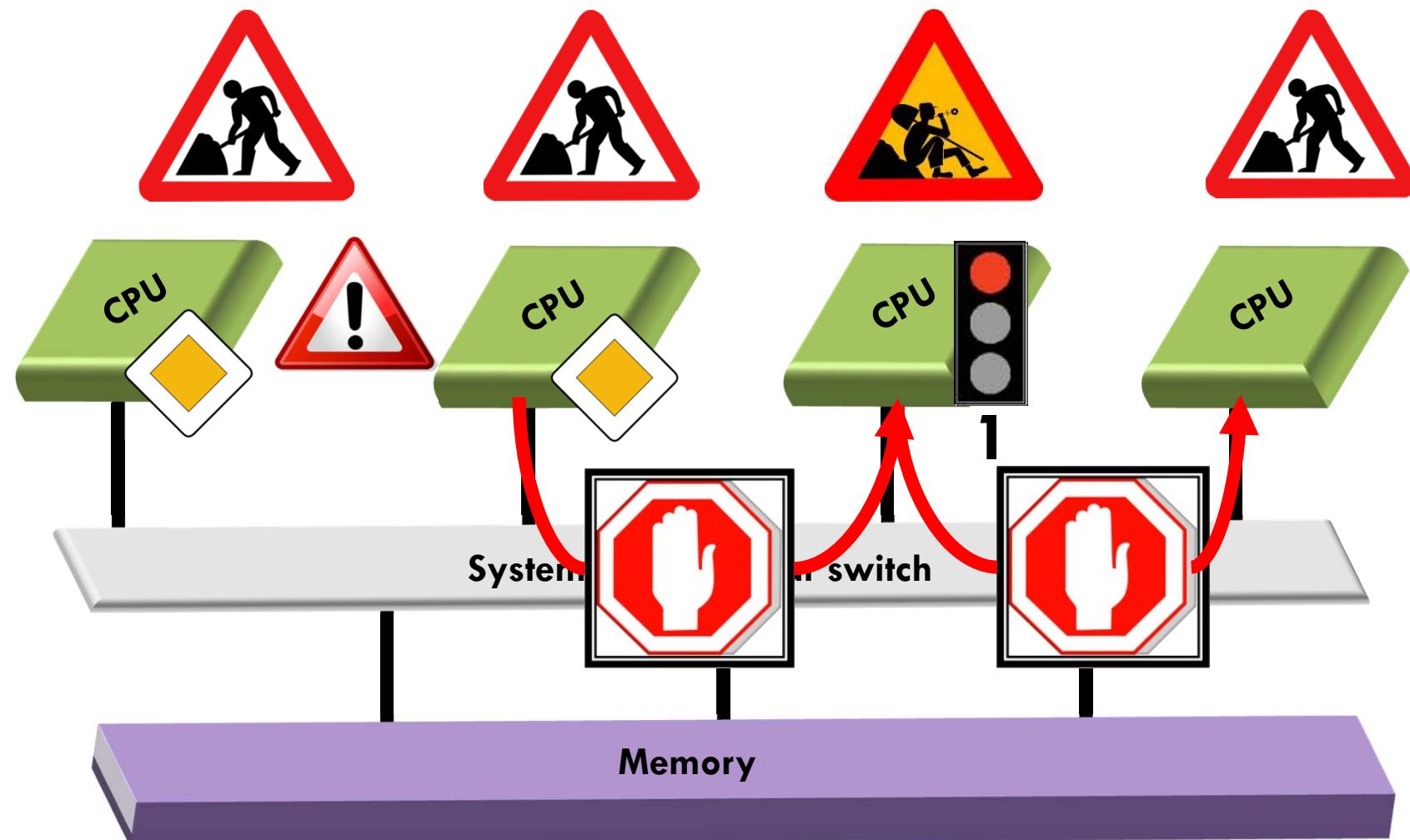
Switching to isolation: One request



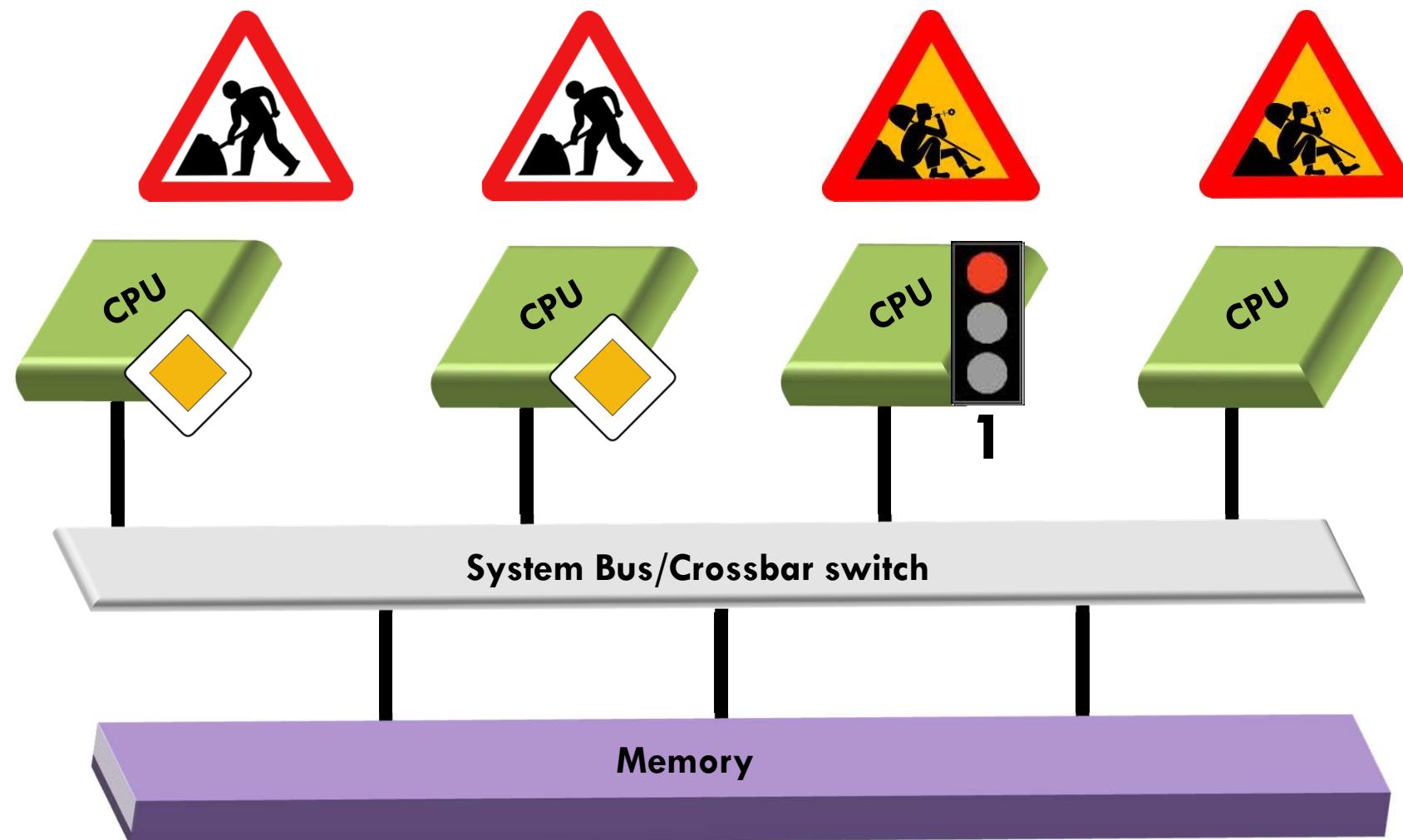
Switching to isolation: One request



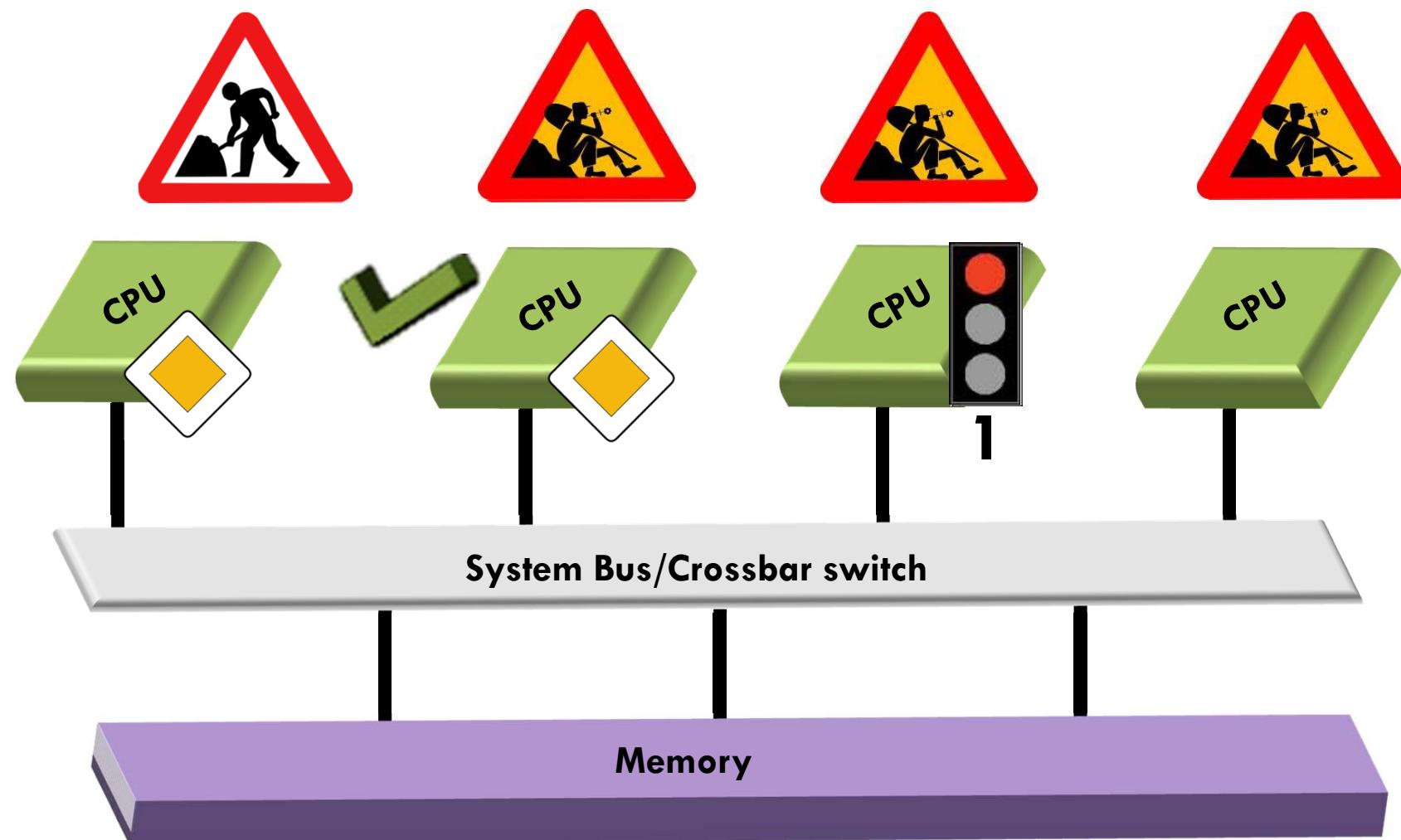
Switching to isolation: One request



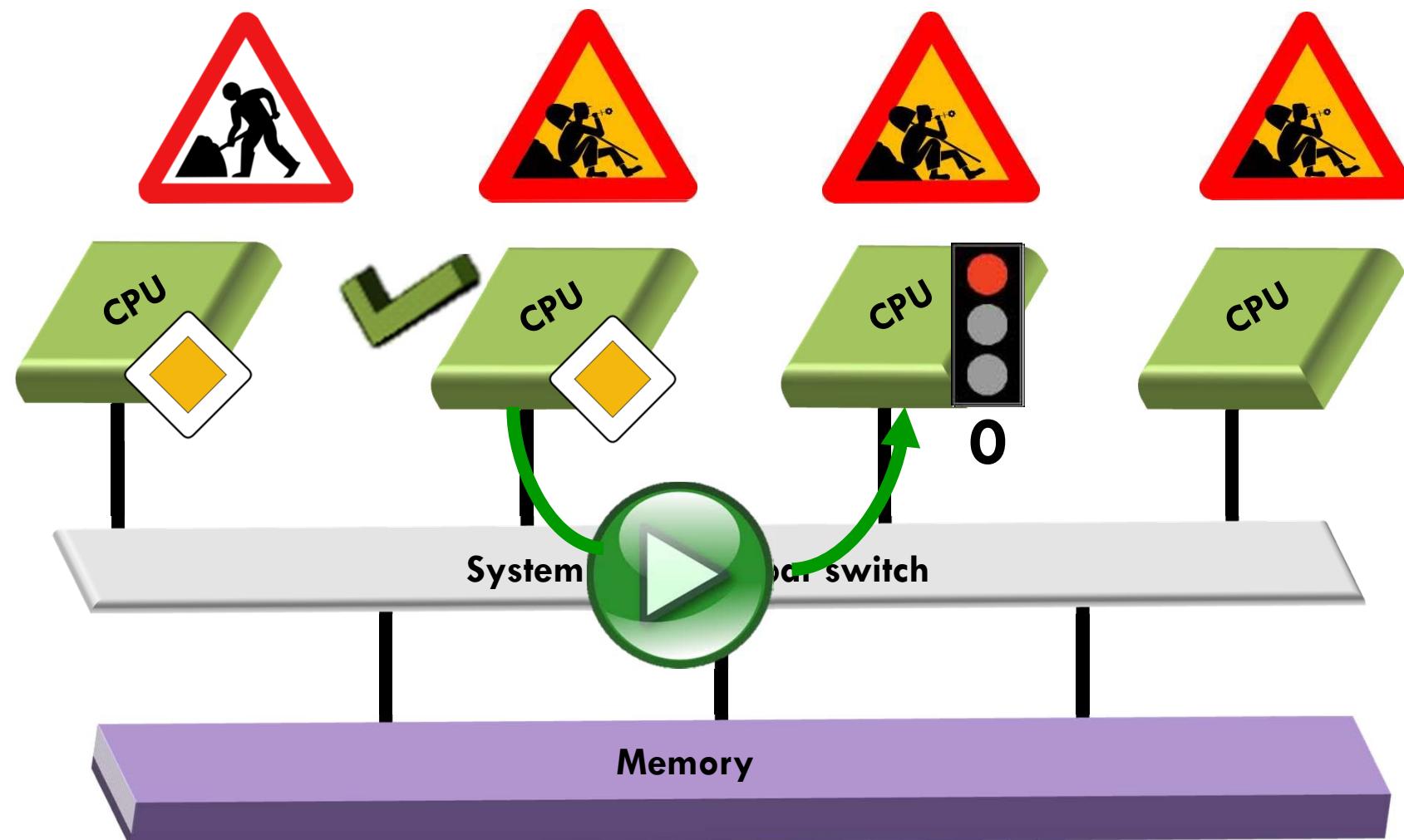
Switching to isolation: One request



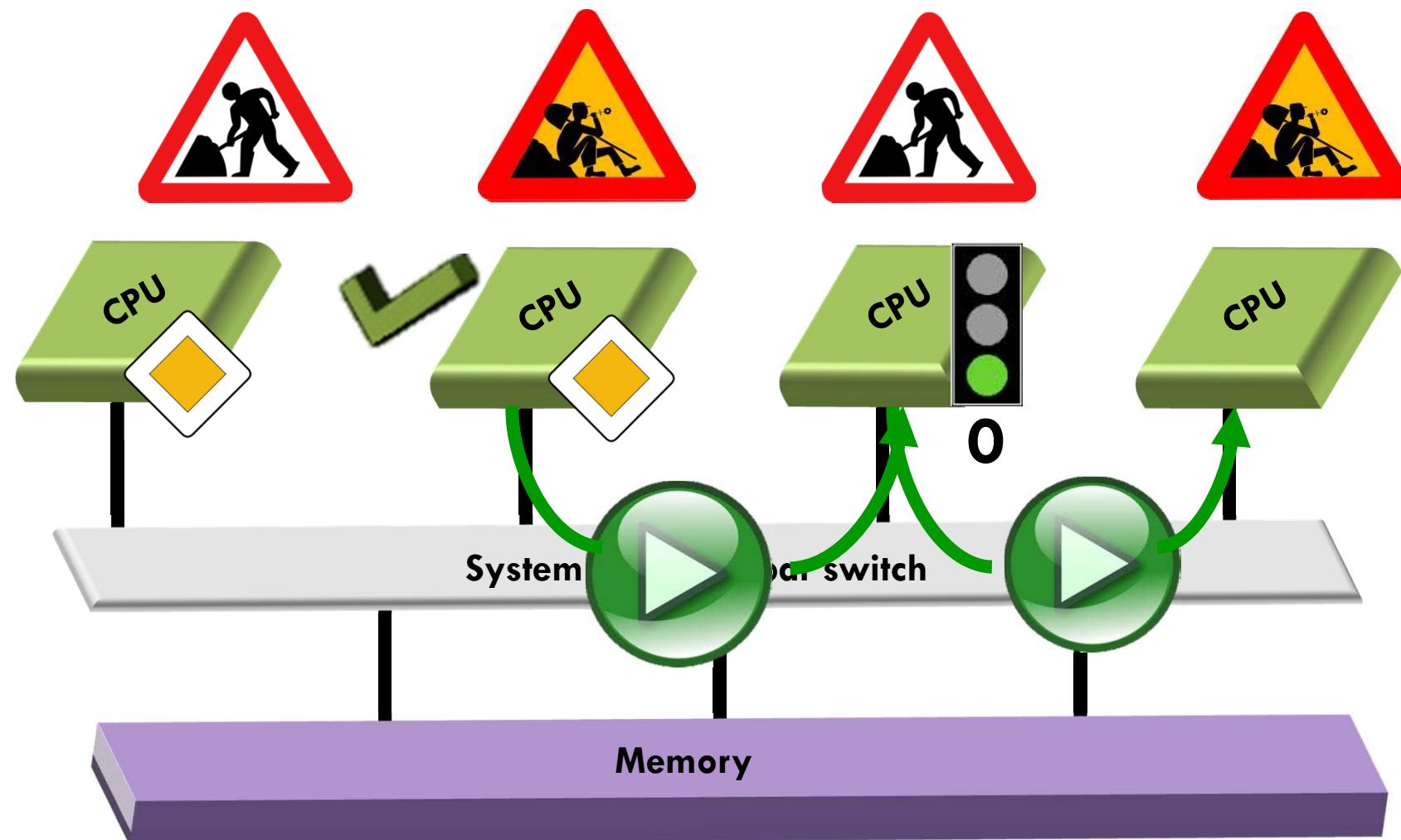
Switching to isolation: One request



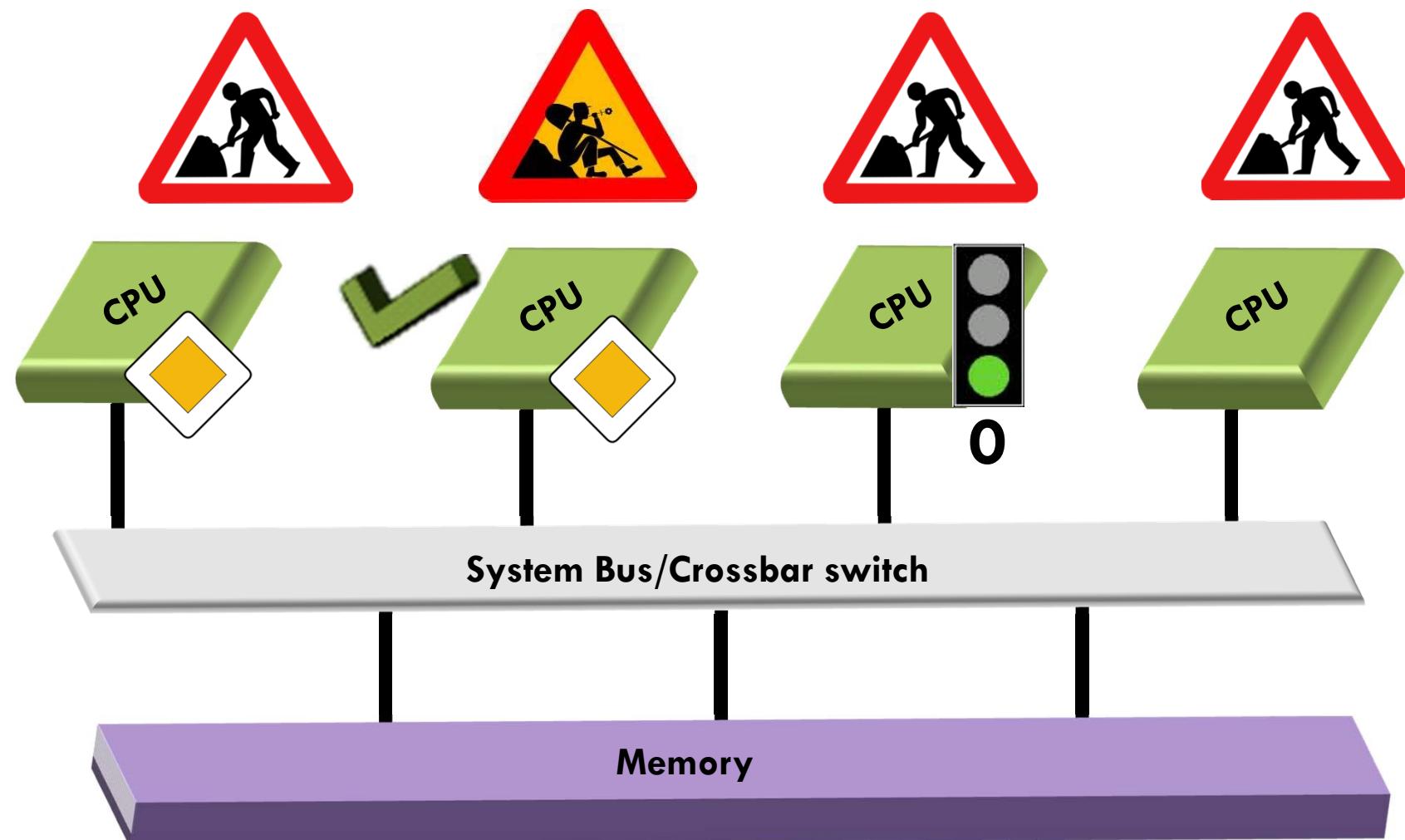
Switching to isolation: One request



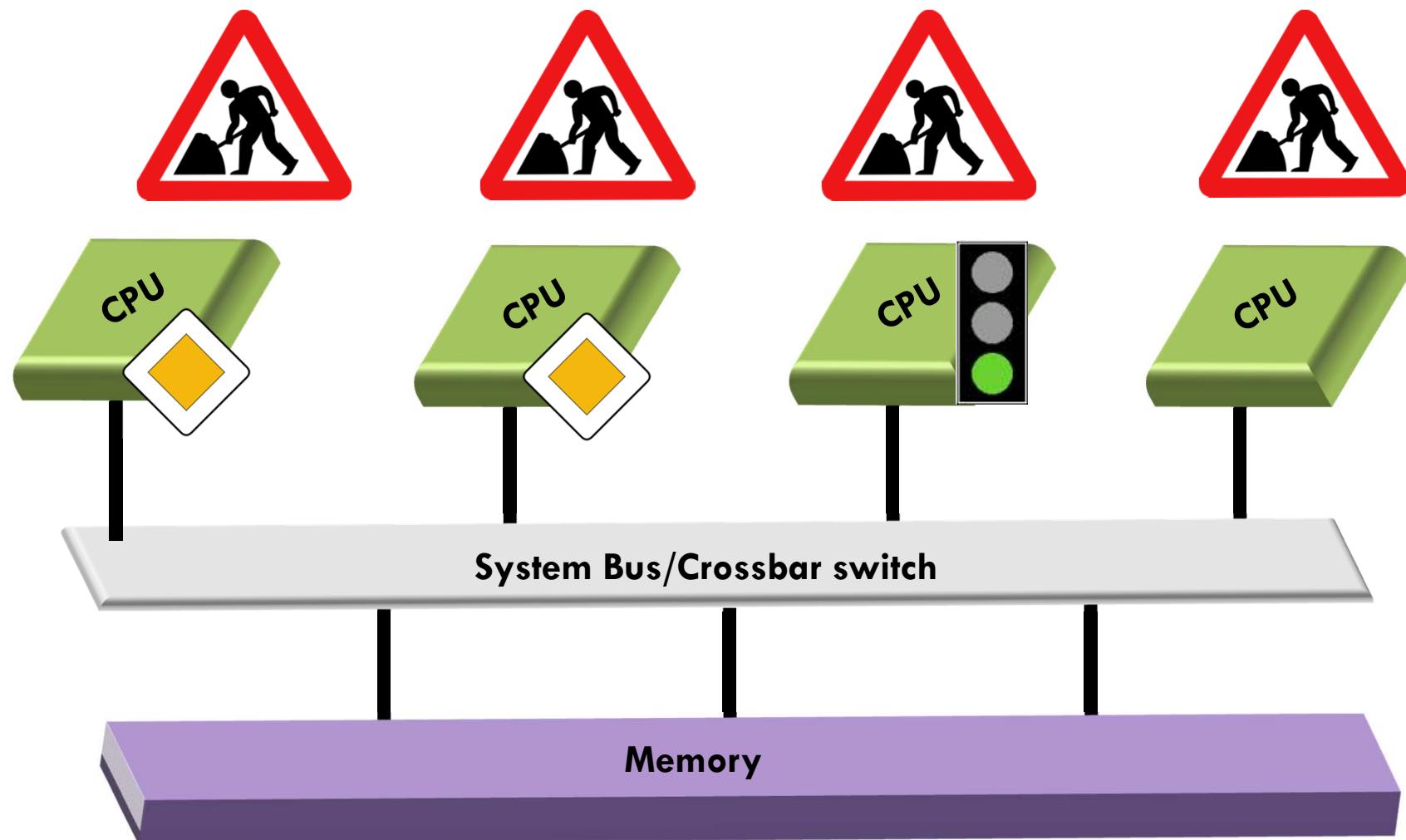
Switching to isolation: One request



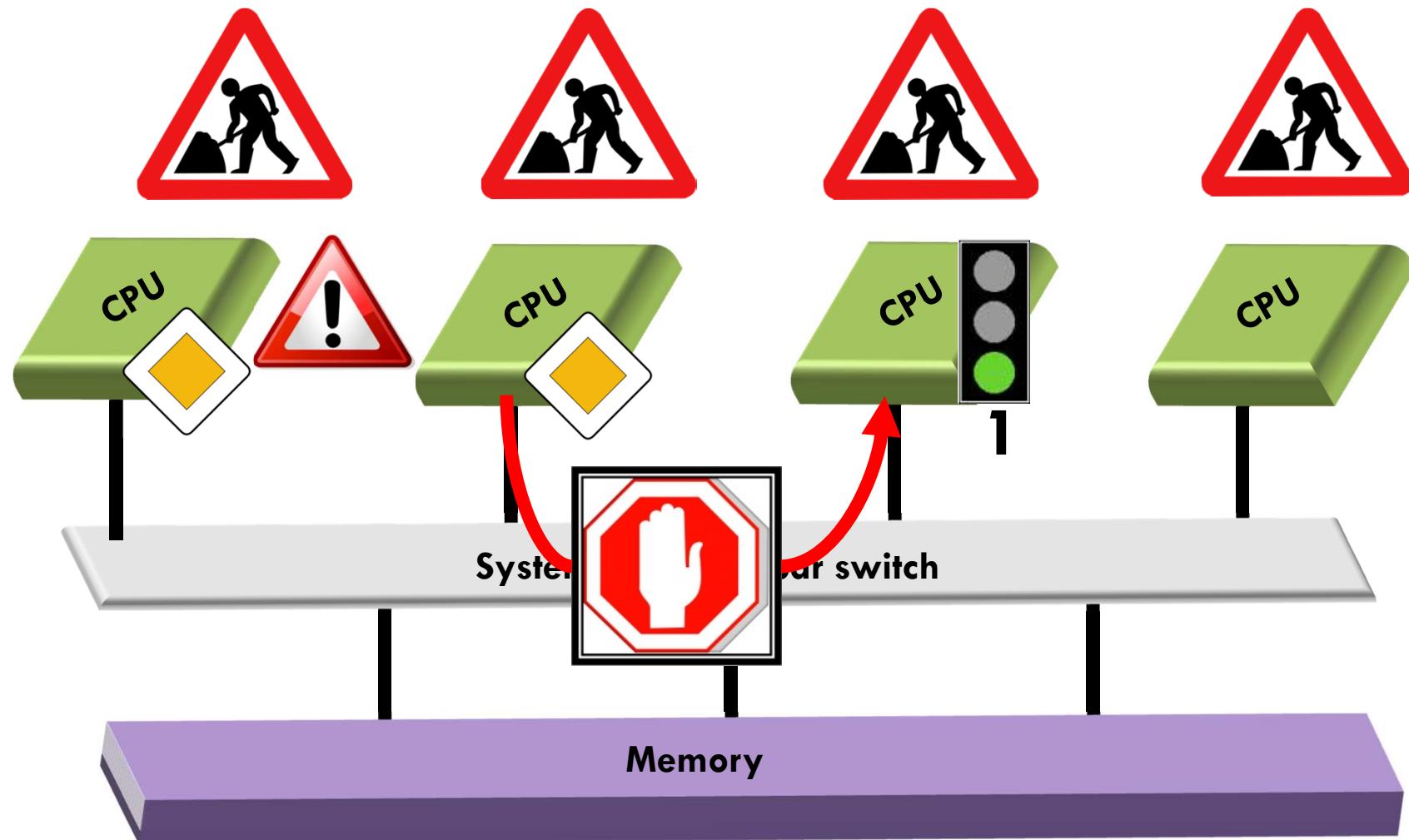
Switching to isolation: One request



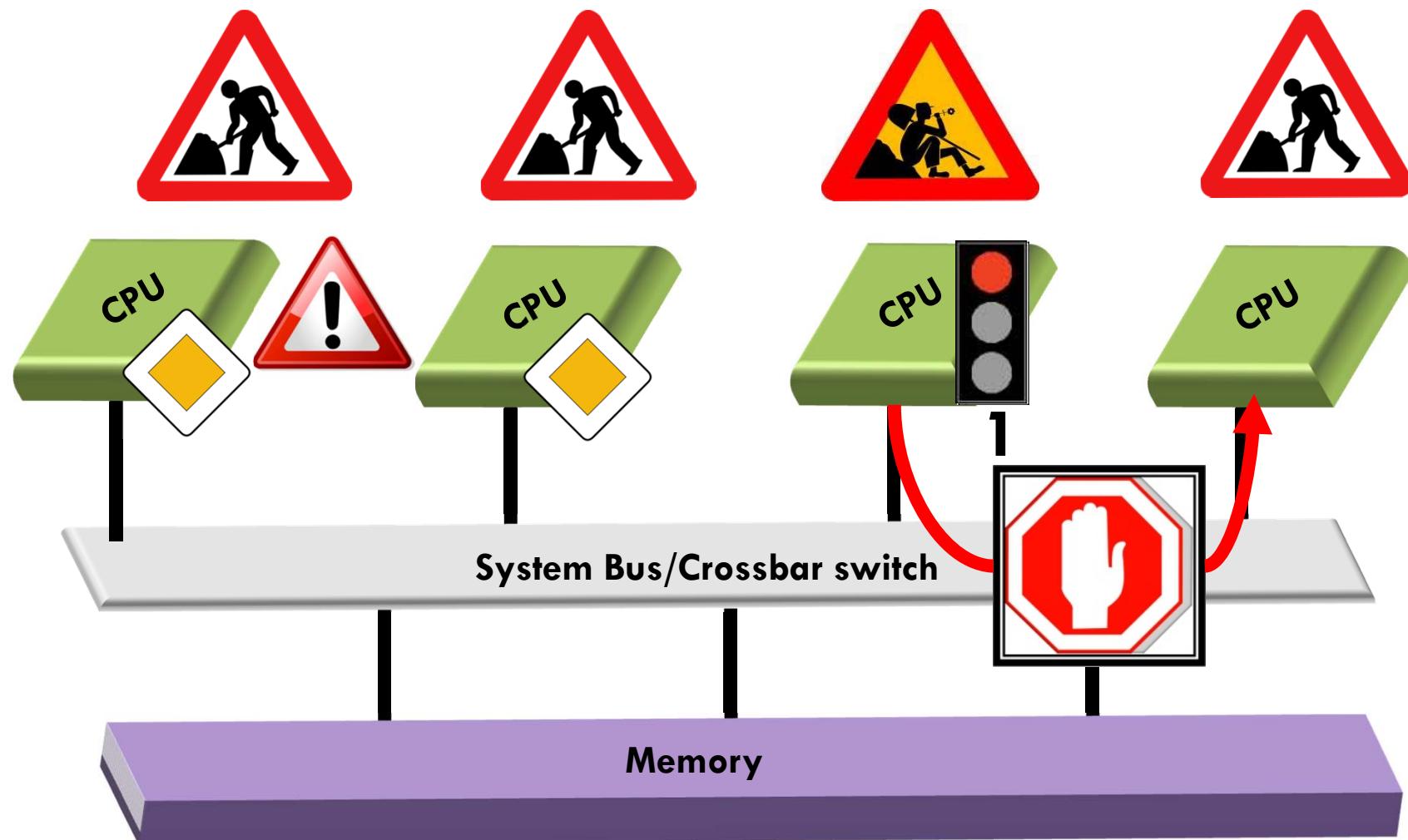
Switching to isolation: Several requests



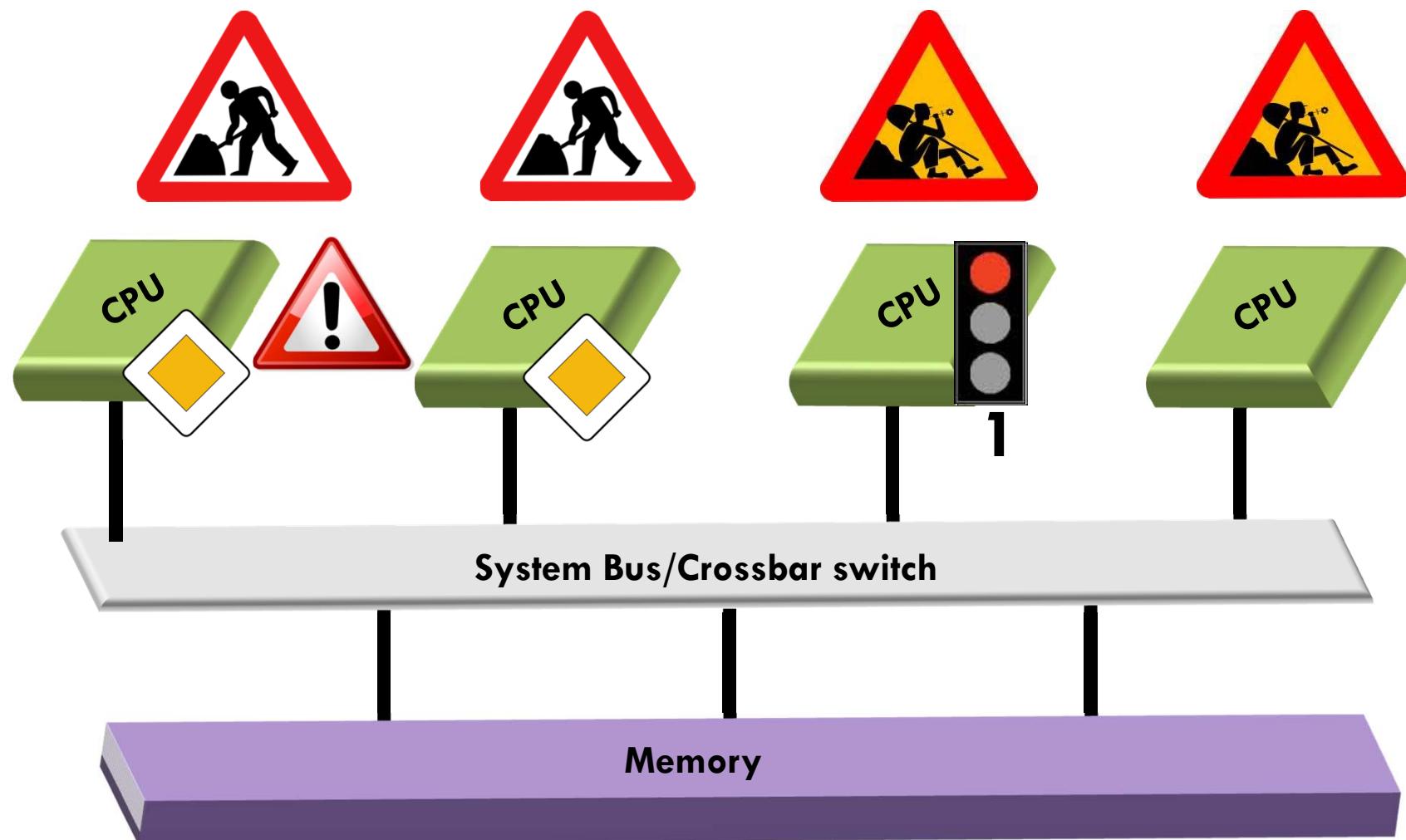
Switching to isolation: Several requests



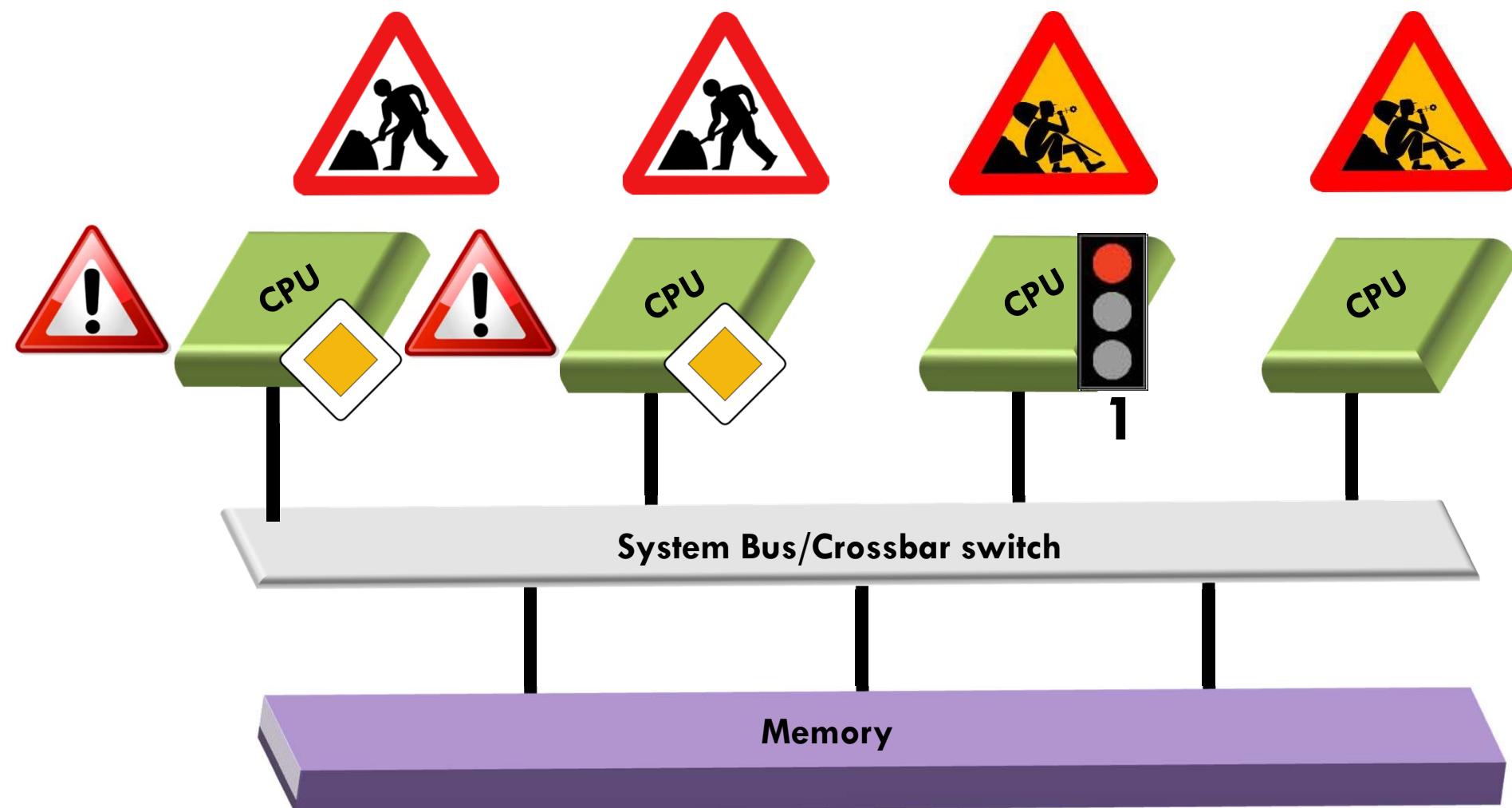
Switching to isolation: Several requests



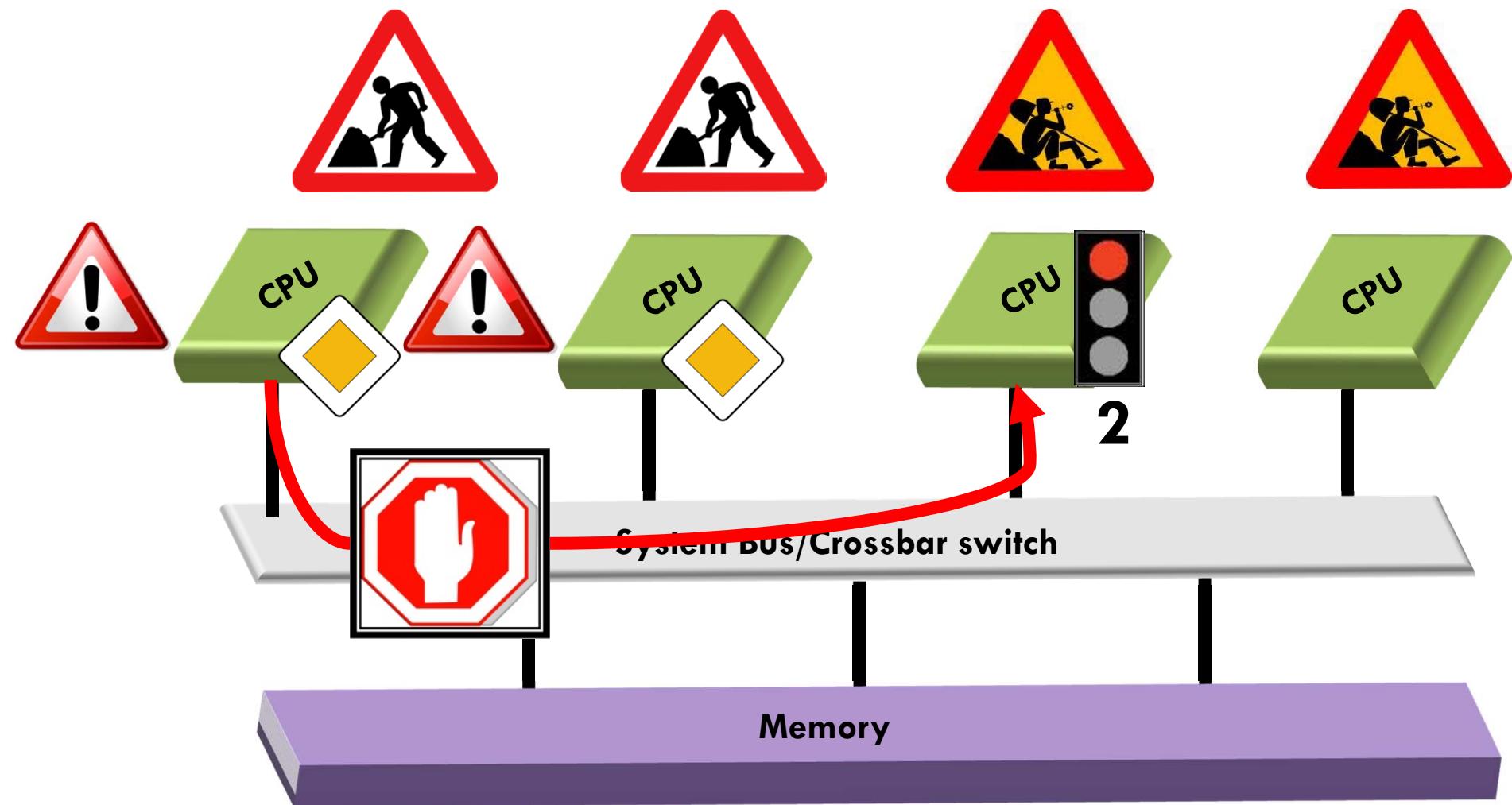
Switching to isolation: Several requests



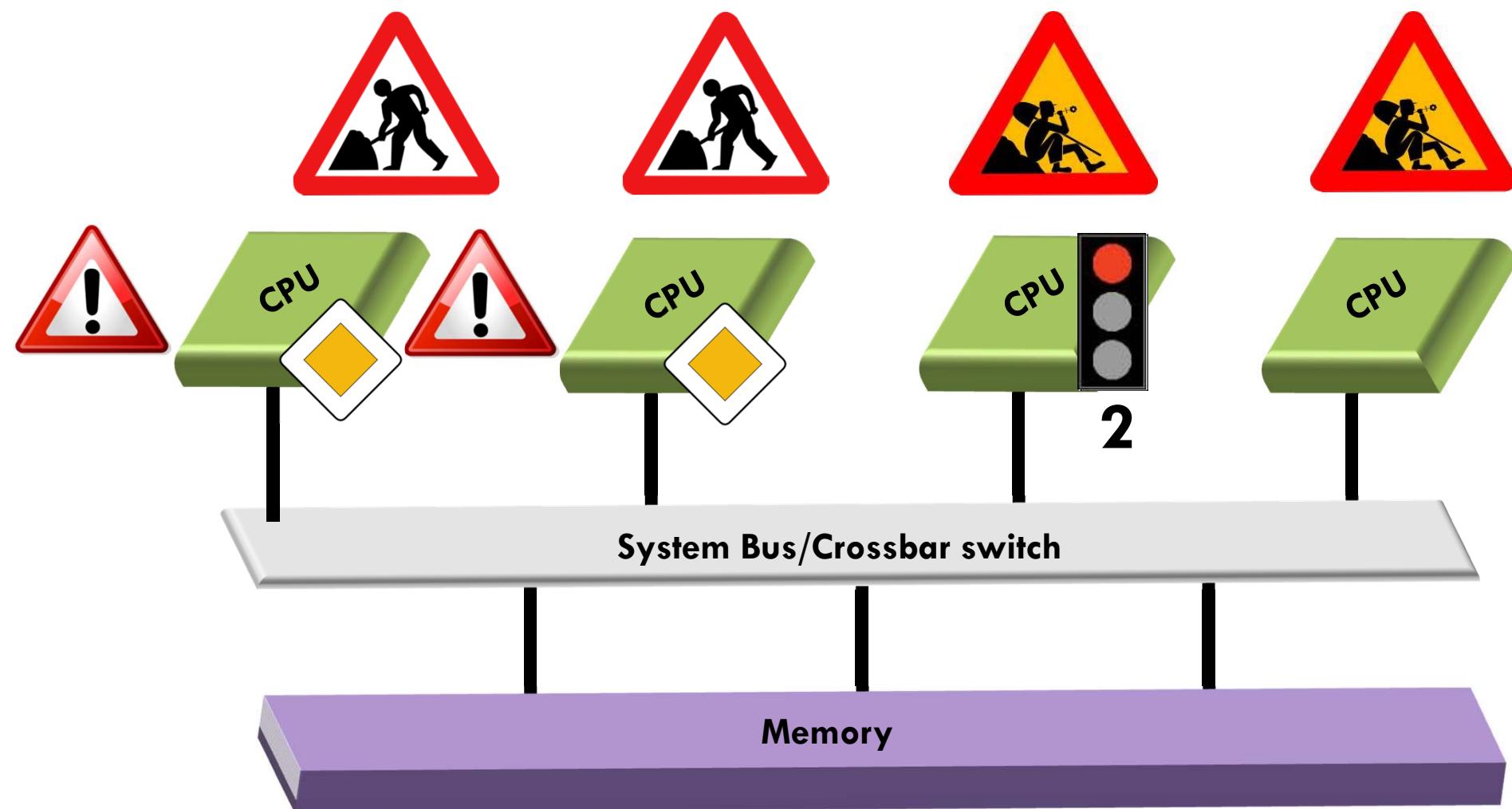
Switching to isolation: Several requests



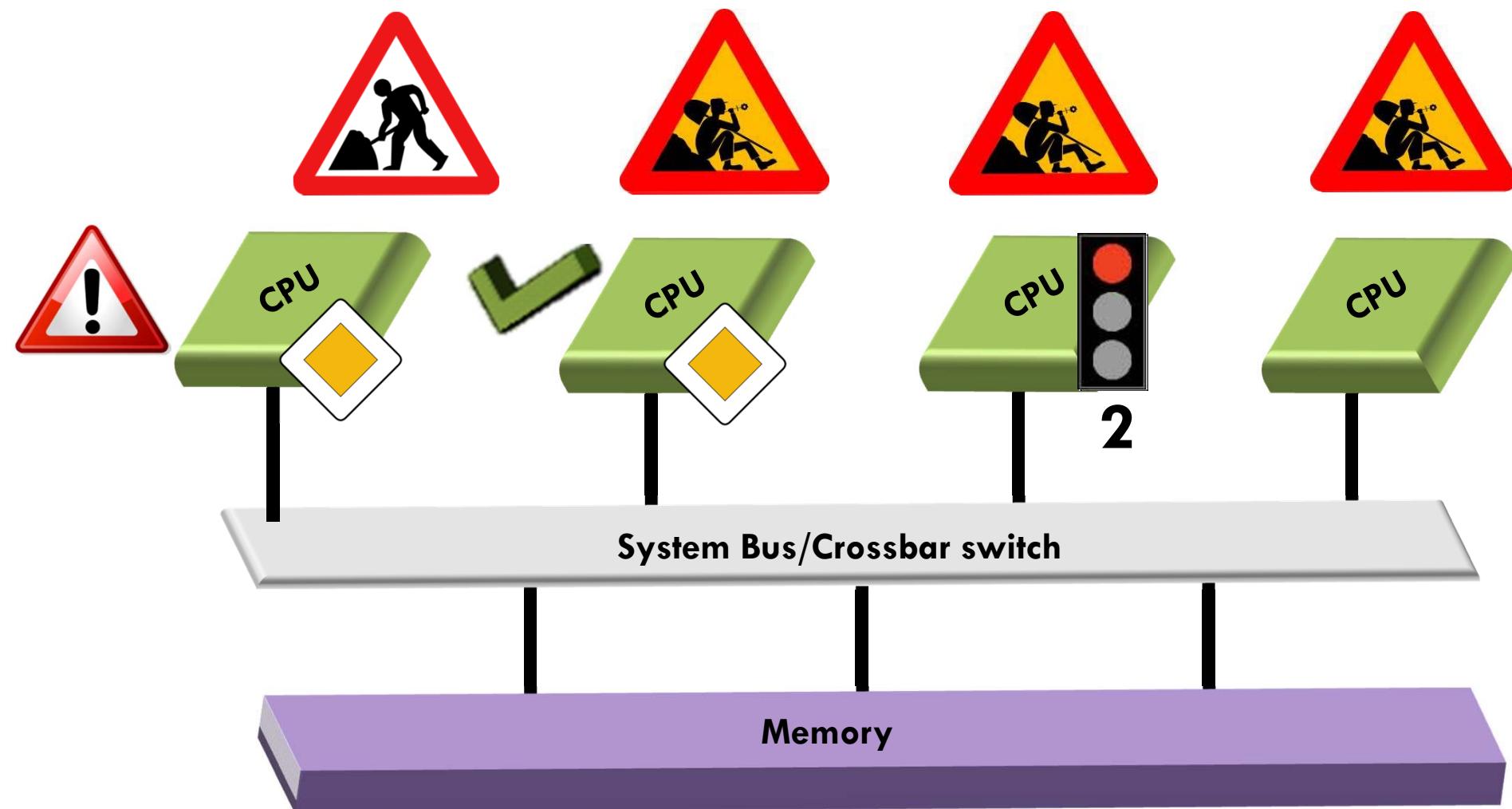
Switching to isolation: Several requests



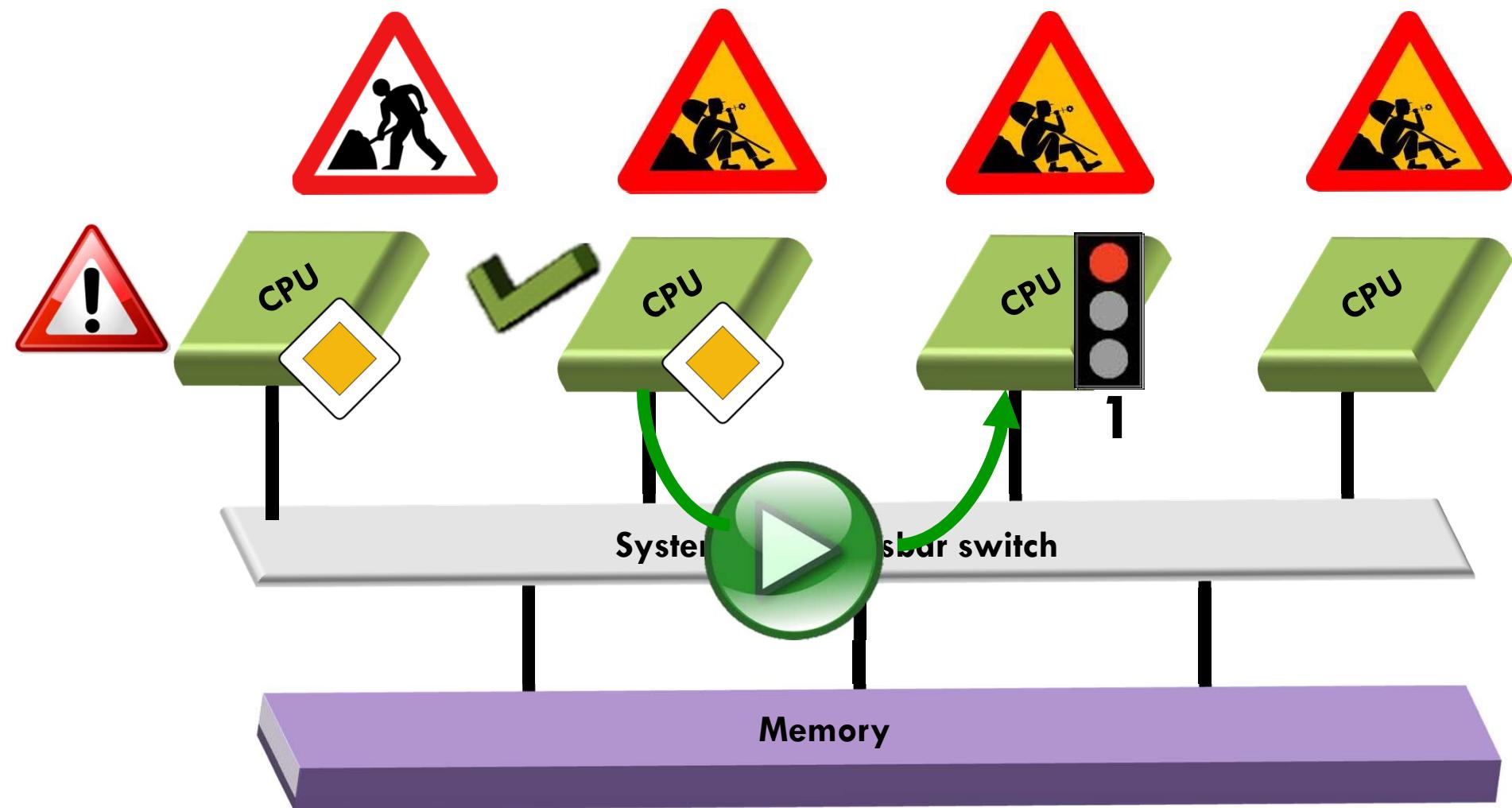
Switching to isolation: Several requests



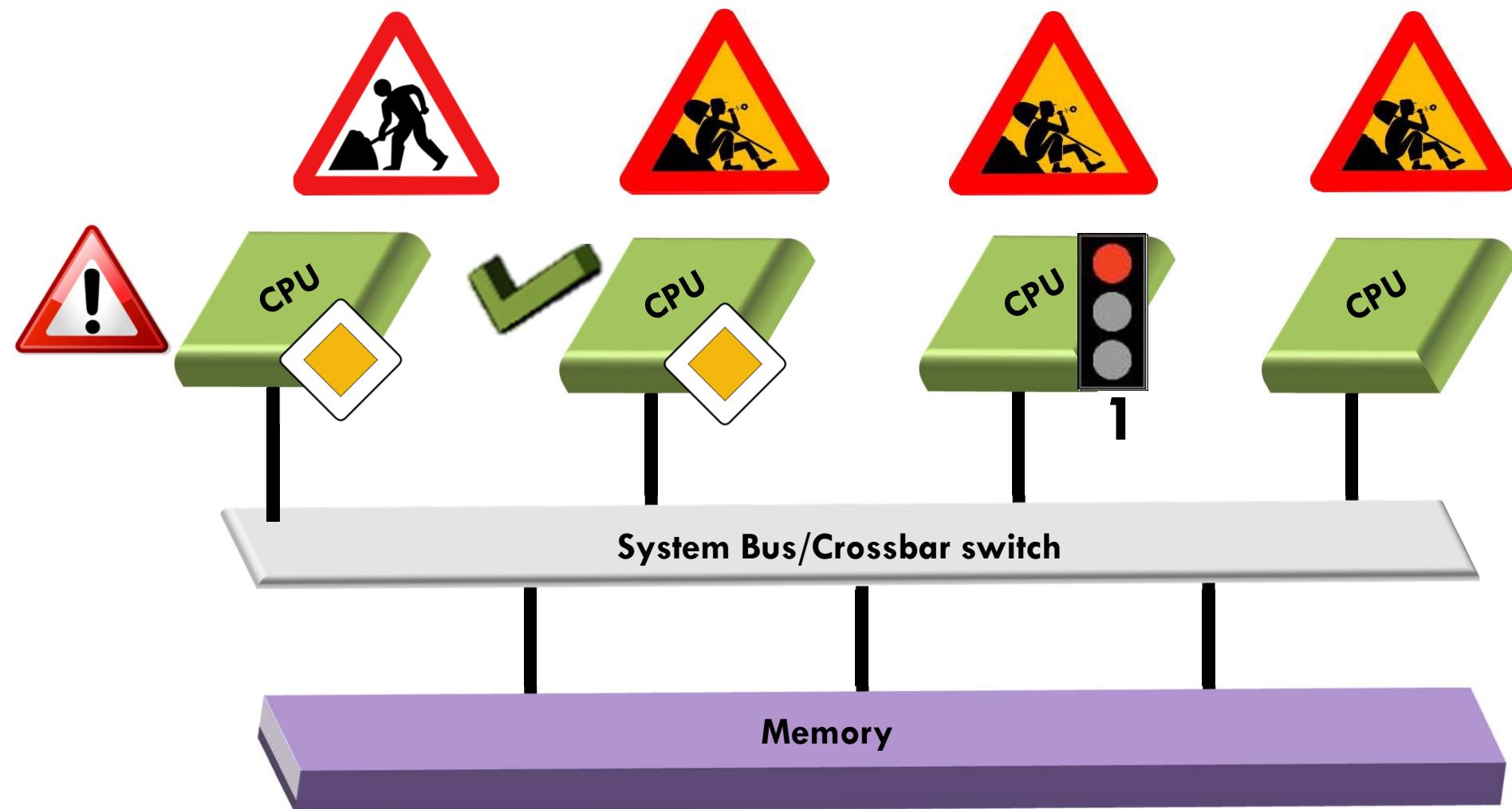
Switching to isolation: Several requests



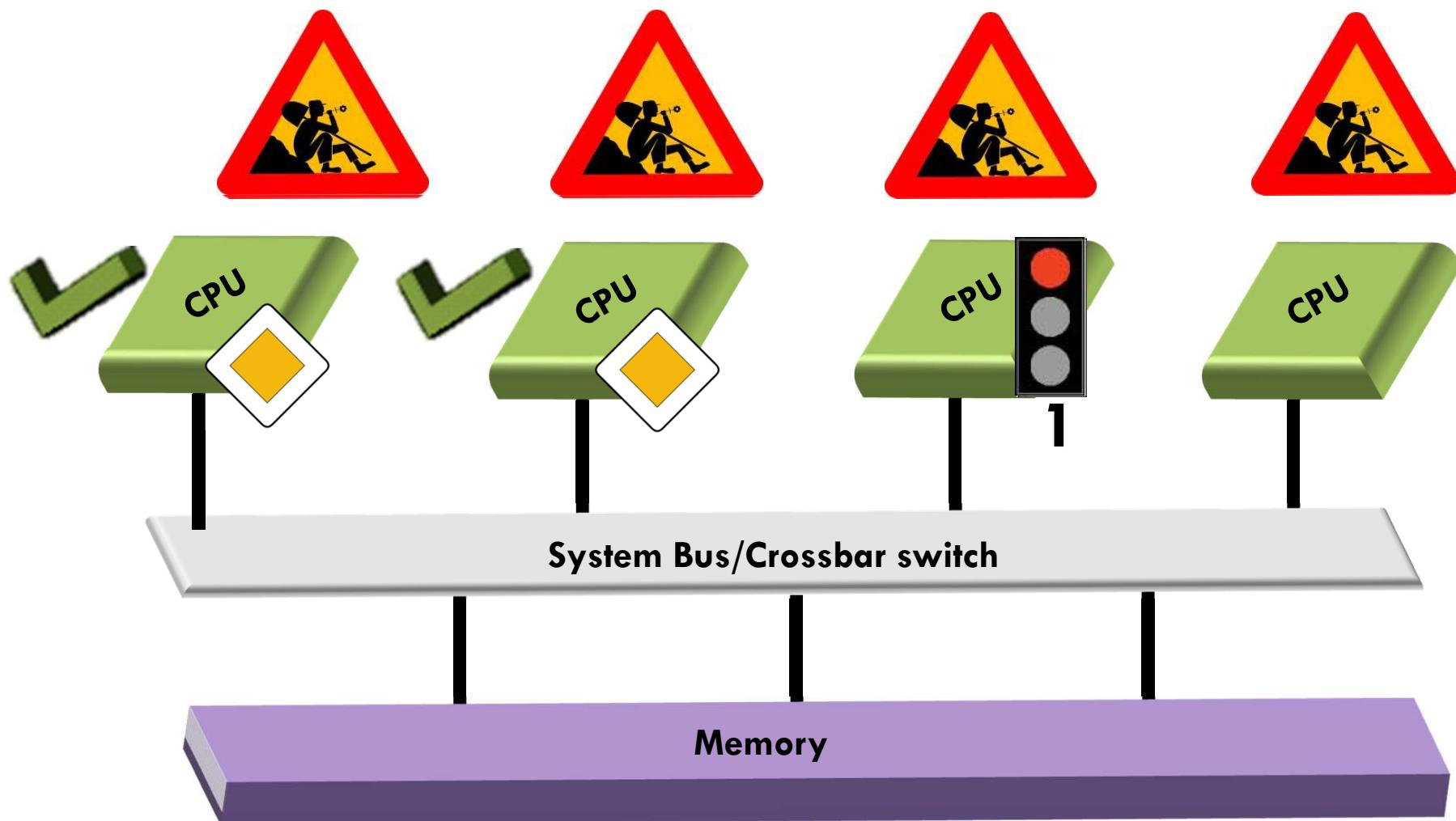
Switching to isolation: Several requests



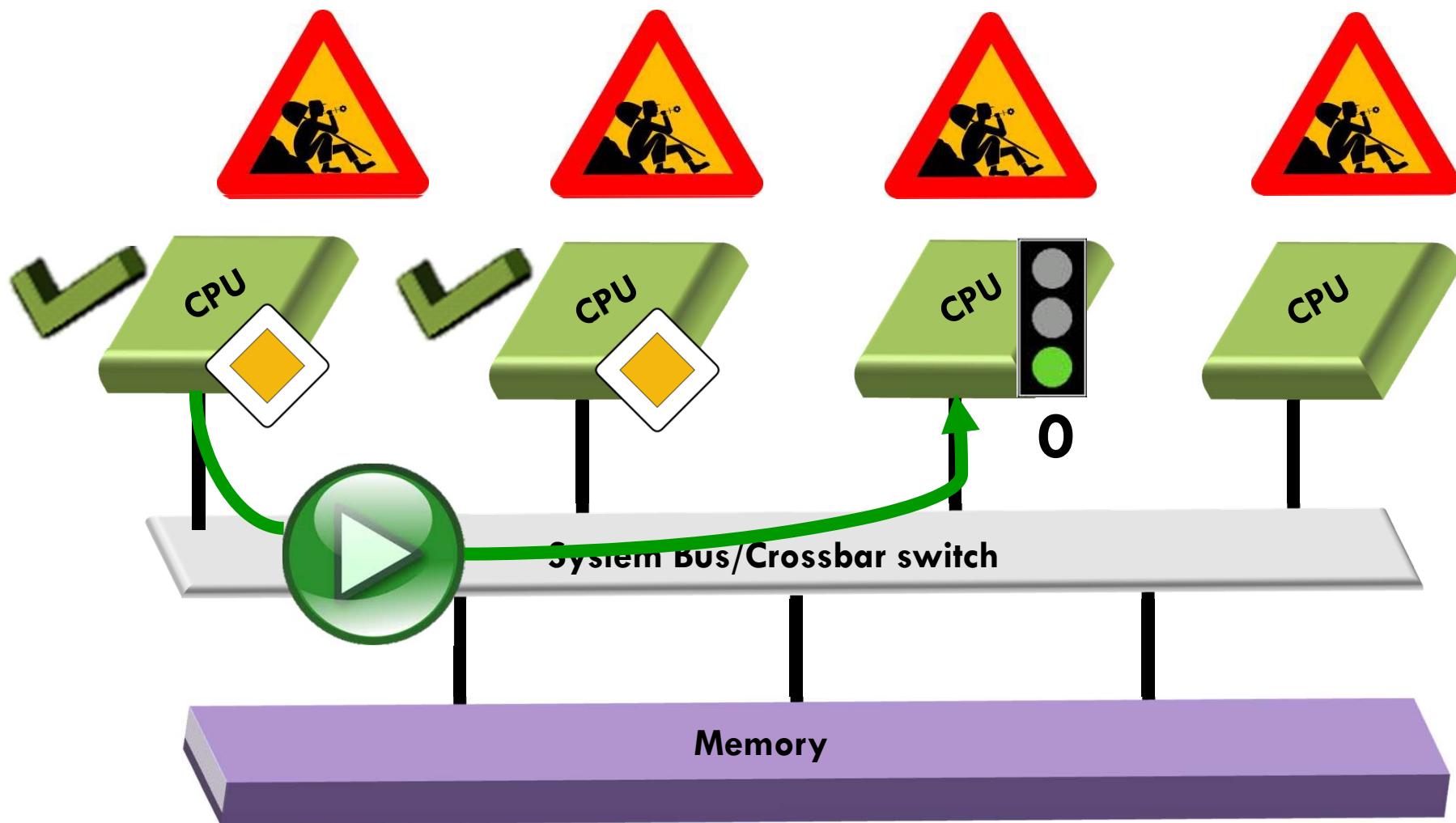
Switching to isolation: Several requests



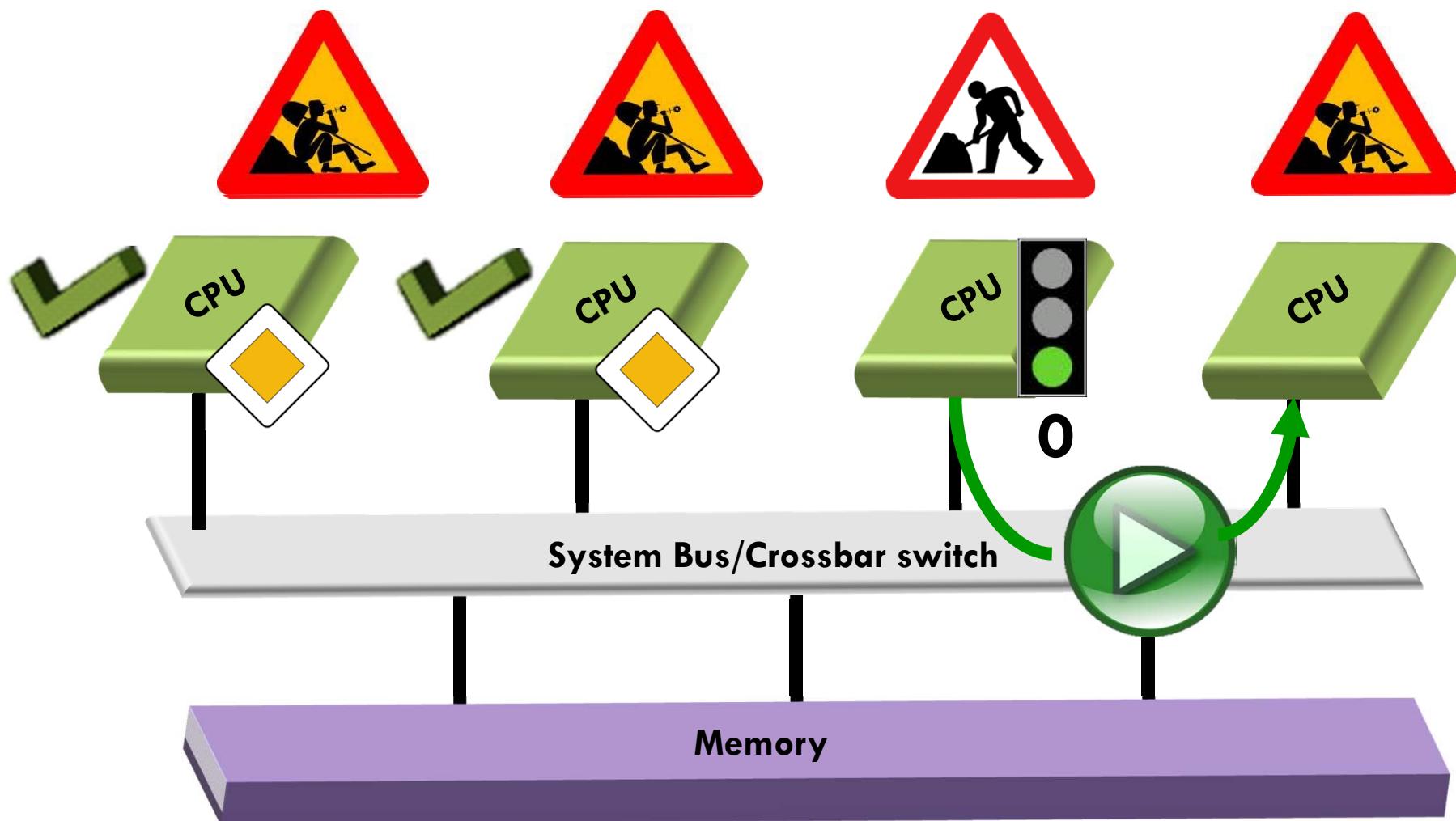
Switching to isolation: Several requests



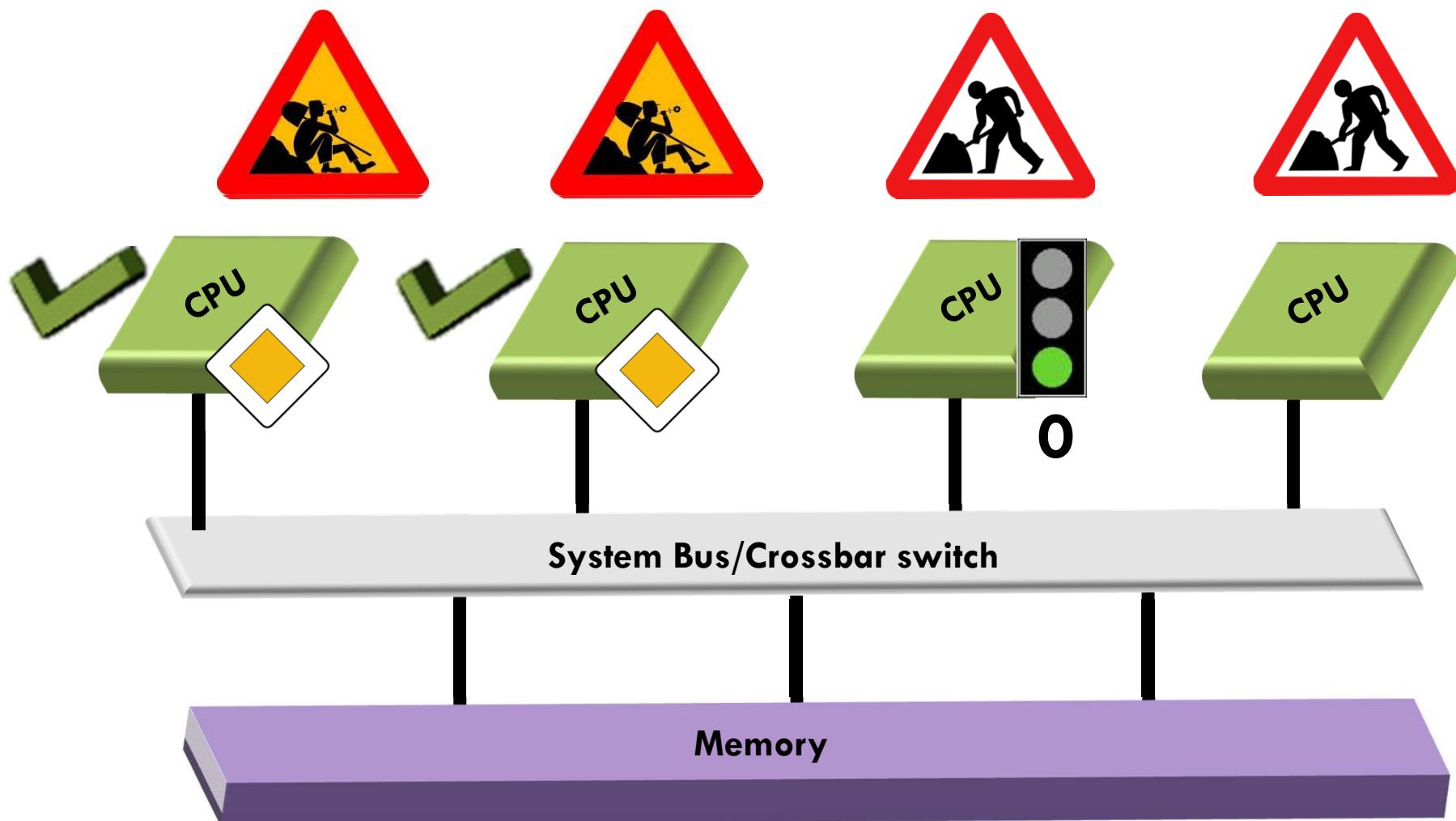
Switching to isolation: Several requests



Switching to isolation: Several requests

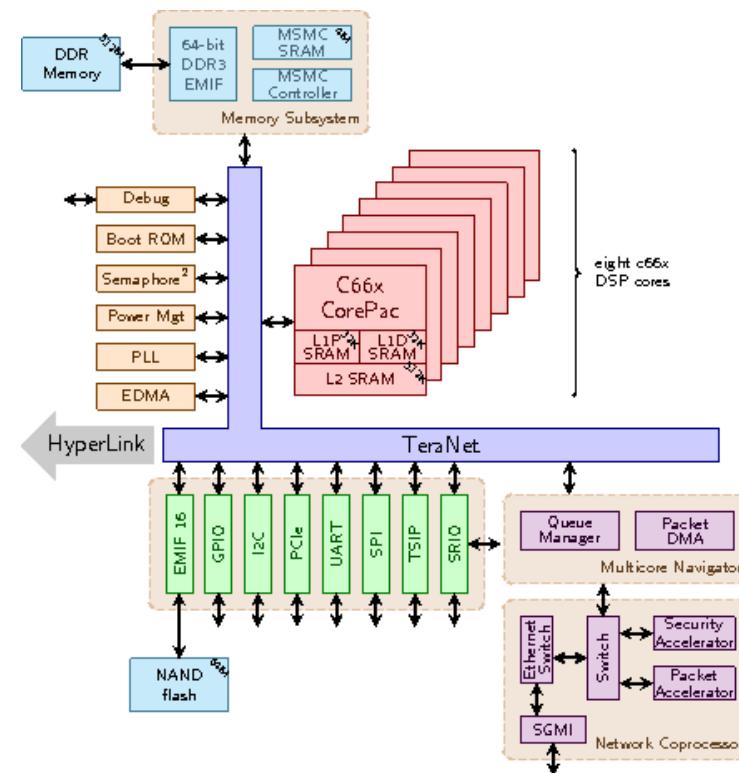


Switching to isolation: Several requests



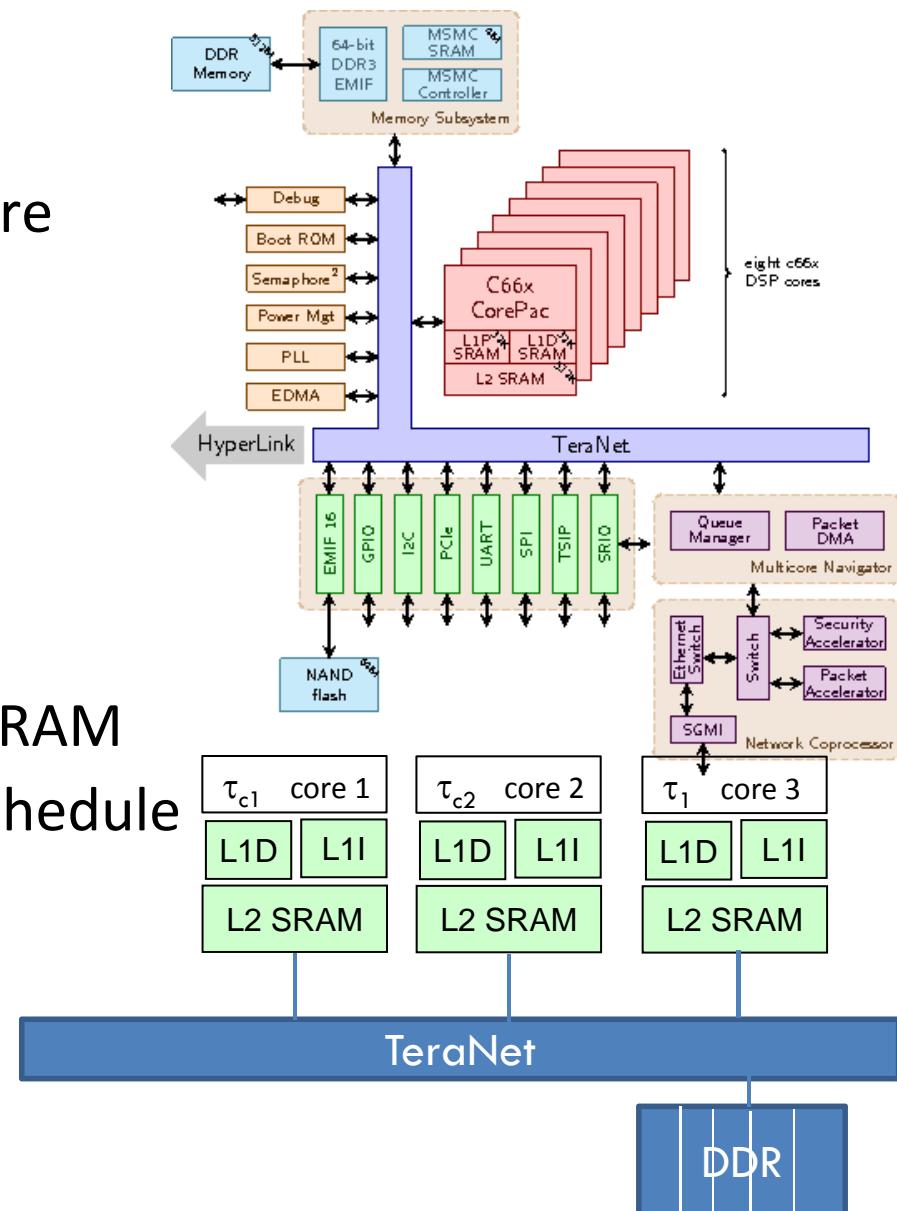
Platform: Texas Instruments TMS320C6678

- 8 TMS320C66x DSP processors
 - clocked at 1 GHz
 - VLIW instruction set architecture
 - 32 KB program memory (L1P)
 - 32 KB data memory (L1D)
 - 512 KB level 2 memory (L2)



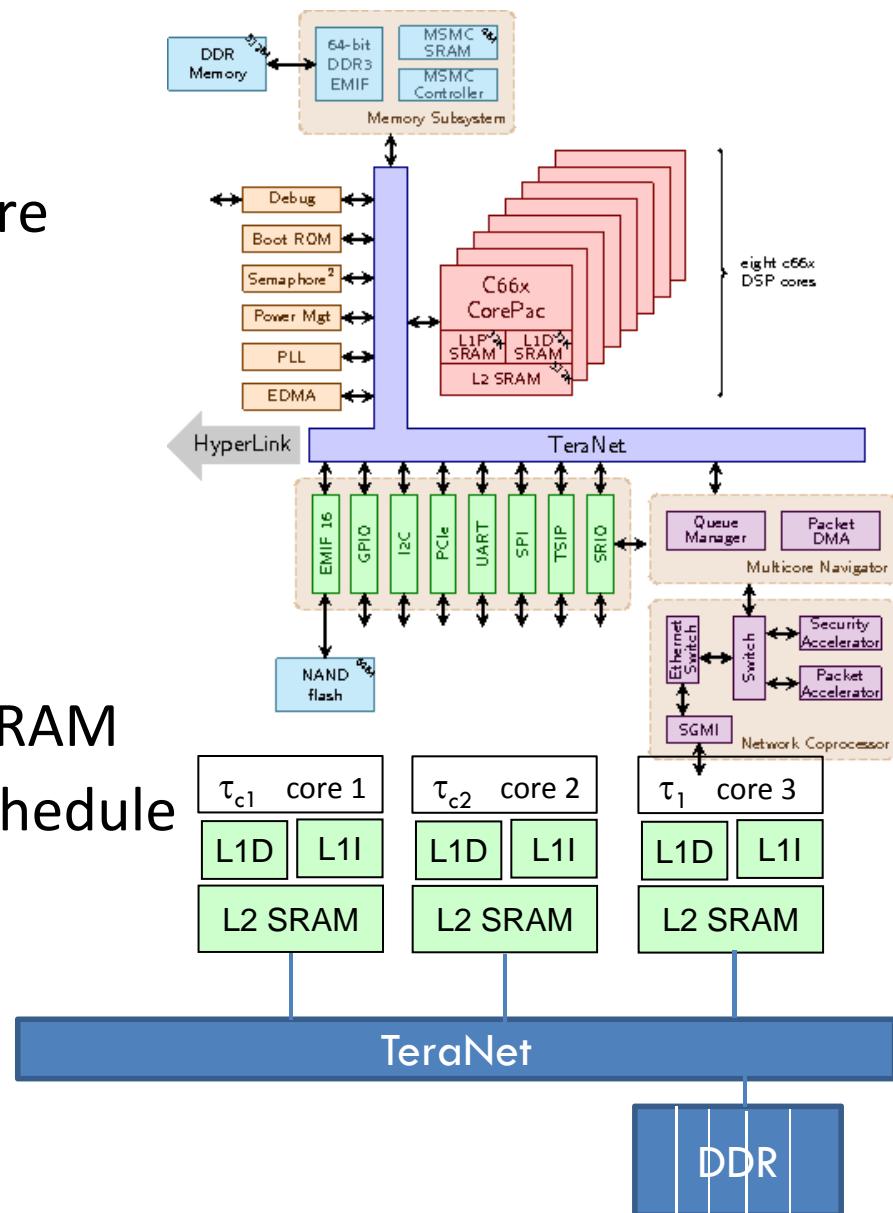
Platform: Texas Instruments TMS320C6678

- 8 TMS320C66x DSP processors
 - clocked at 1 GHz
 - VLIW instruction set architecture
 - 32 KB program memory (L1P)
 - 32 KB data memory (L1D)
 - 512 KB level 2 memory (L2)
- Platform configuration
 - Caches → SRAM
 - Data @ DDR, otherwise @ L2 SRAM
 - Partitioned non pre-emptive schedule



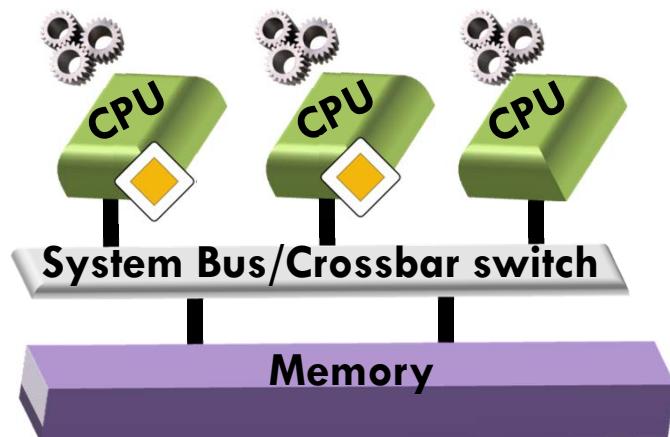
Platform: Texas Instruments TMS320C6678

- 8 TMS320C66x DSP processors
 - clocked at 1 GHz
 - VLIW instruction set architecture
 - 32 KB program memory (L1P)
 - 32 KB data memory (L1D)
 - 512 KB level 2 memory (L2)
- Platform configuration
 - Caches → SRAM
 - Data @ DDR, otherwise @ L2 SRAM
 - Partitioned non pre-emptive schedule
- Bare-metal library
 - Time management
 - Message passing
 - Events & Interrupts



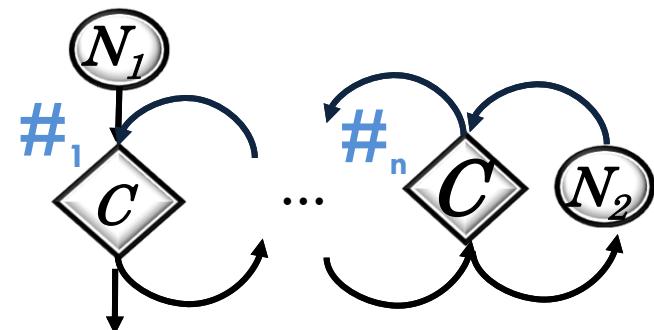
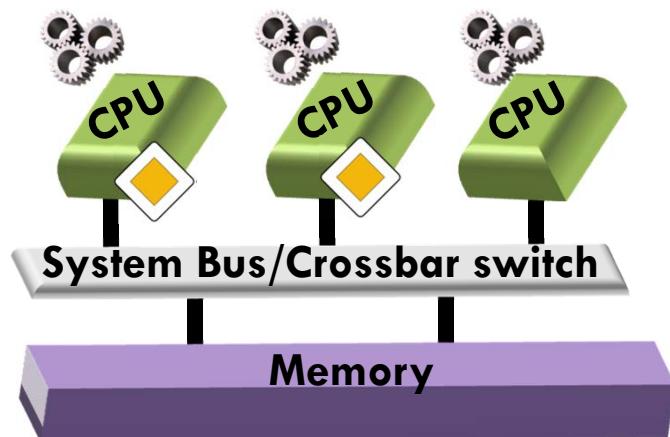
Experimental setup

- Number of parallel tasks
 - 1 - 6 cores running less critical



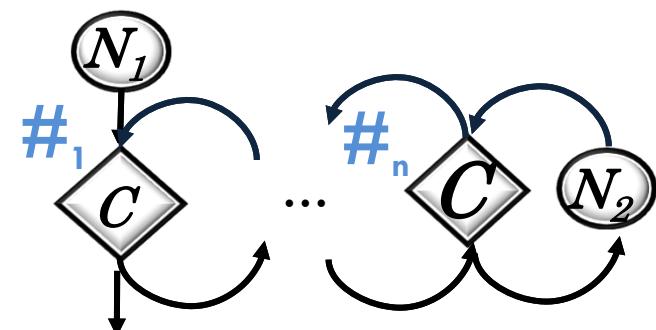
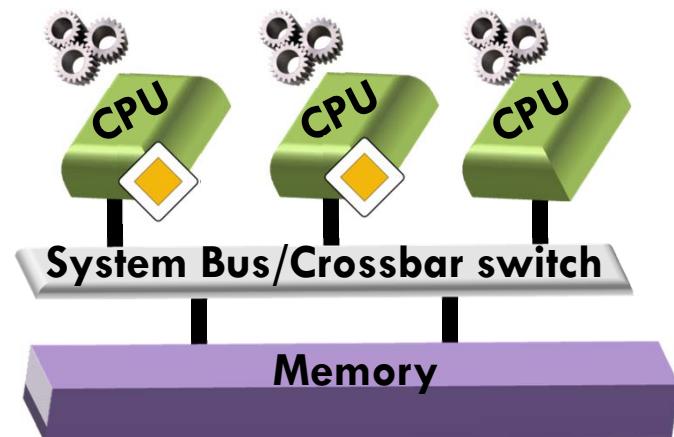
Experimental setup

- Number of parallel tasks
 - 1 - 6 cores running less critical
- Size of critical application
 - Several loop bounds



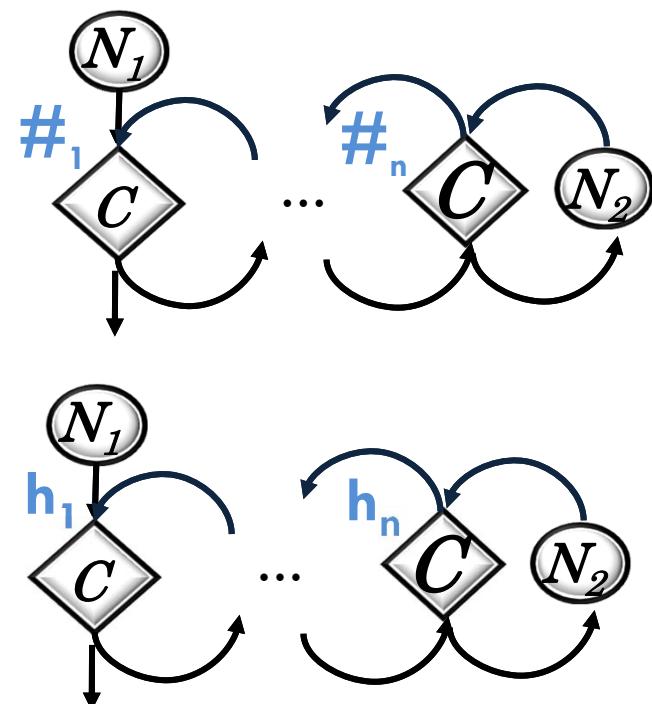
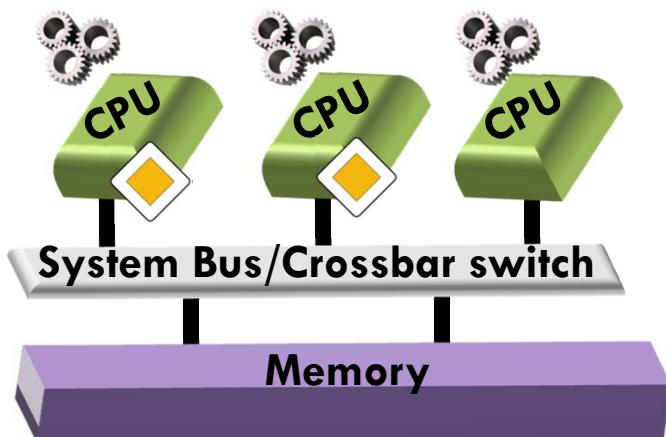
Experimental setup

- Number of parallel tasks
 - 1 - 6 cores running less critical
- Size of critical application
 - Several loop bounds
- Critical deadlines
 - Tight (close to WCET_{iso})
 - More relaxed

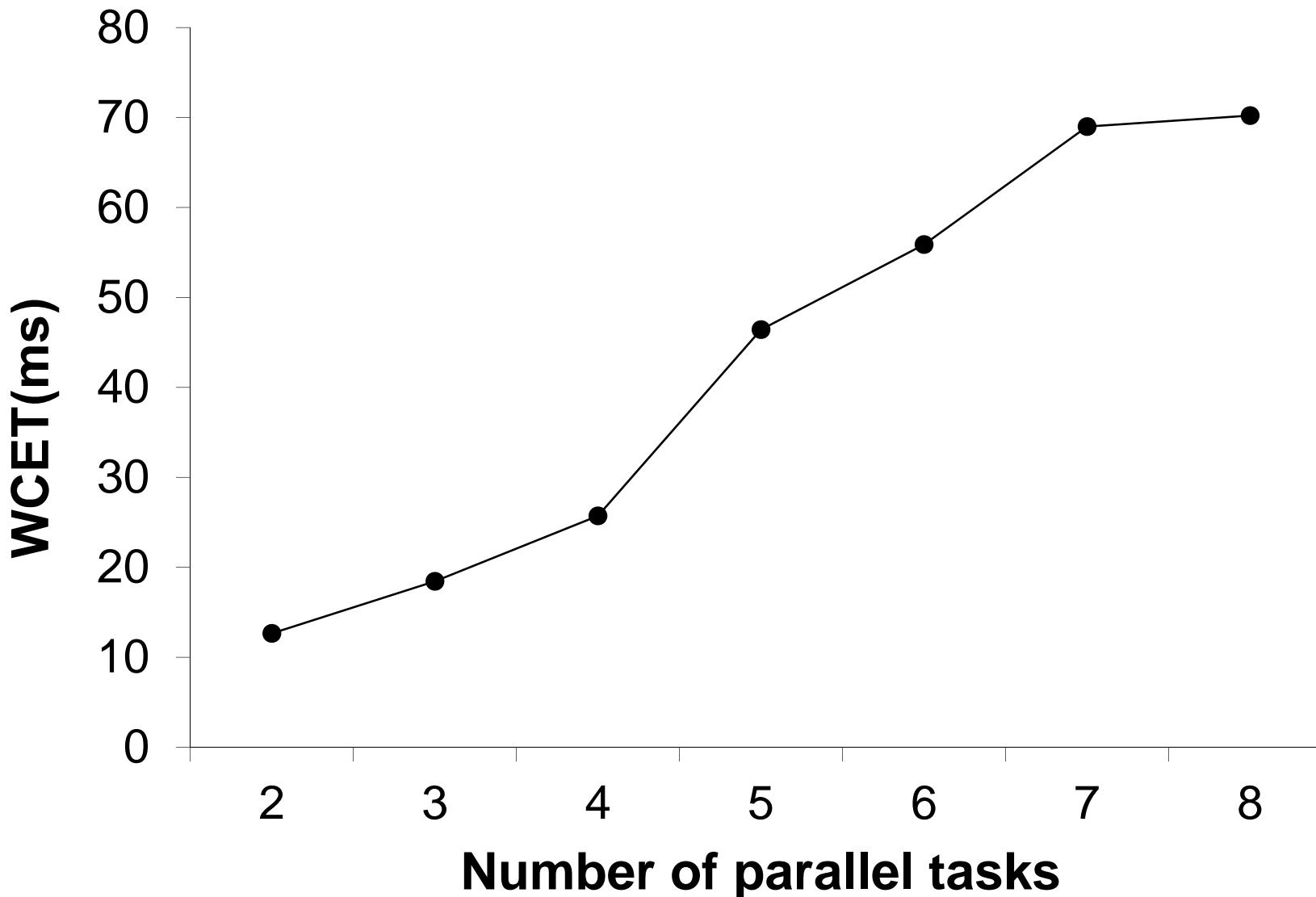


Experimental setup

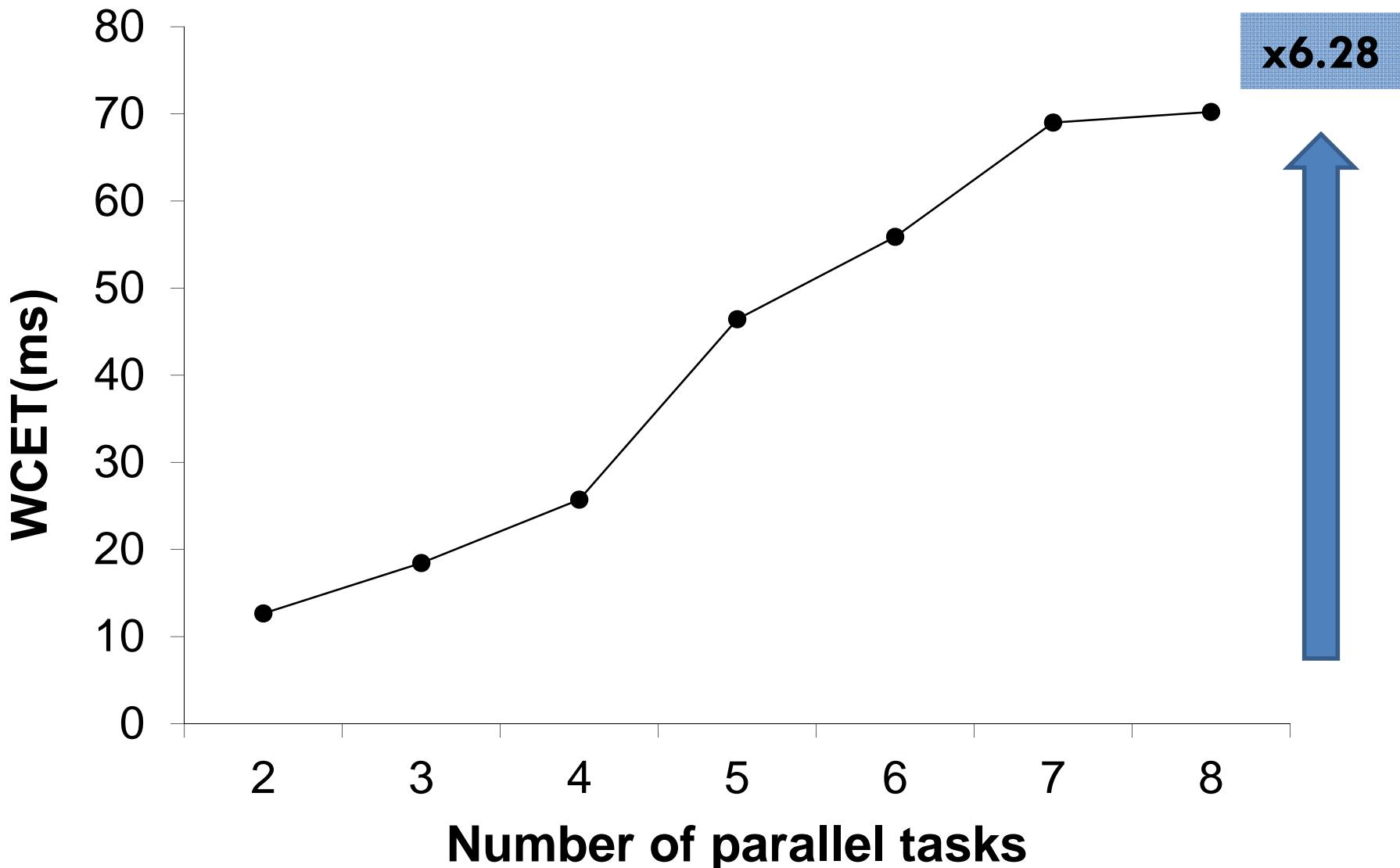
- Number of parallel tasks
 - 1 - 6 cores running less critical
- Size of critical application
 - Several loop bounds
- Critical deadlines
 - Tight (close to WCET_{iso})
 - More relaxed
- Observation points
 - From external nested level
 - To all nested levels



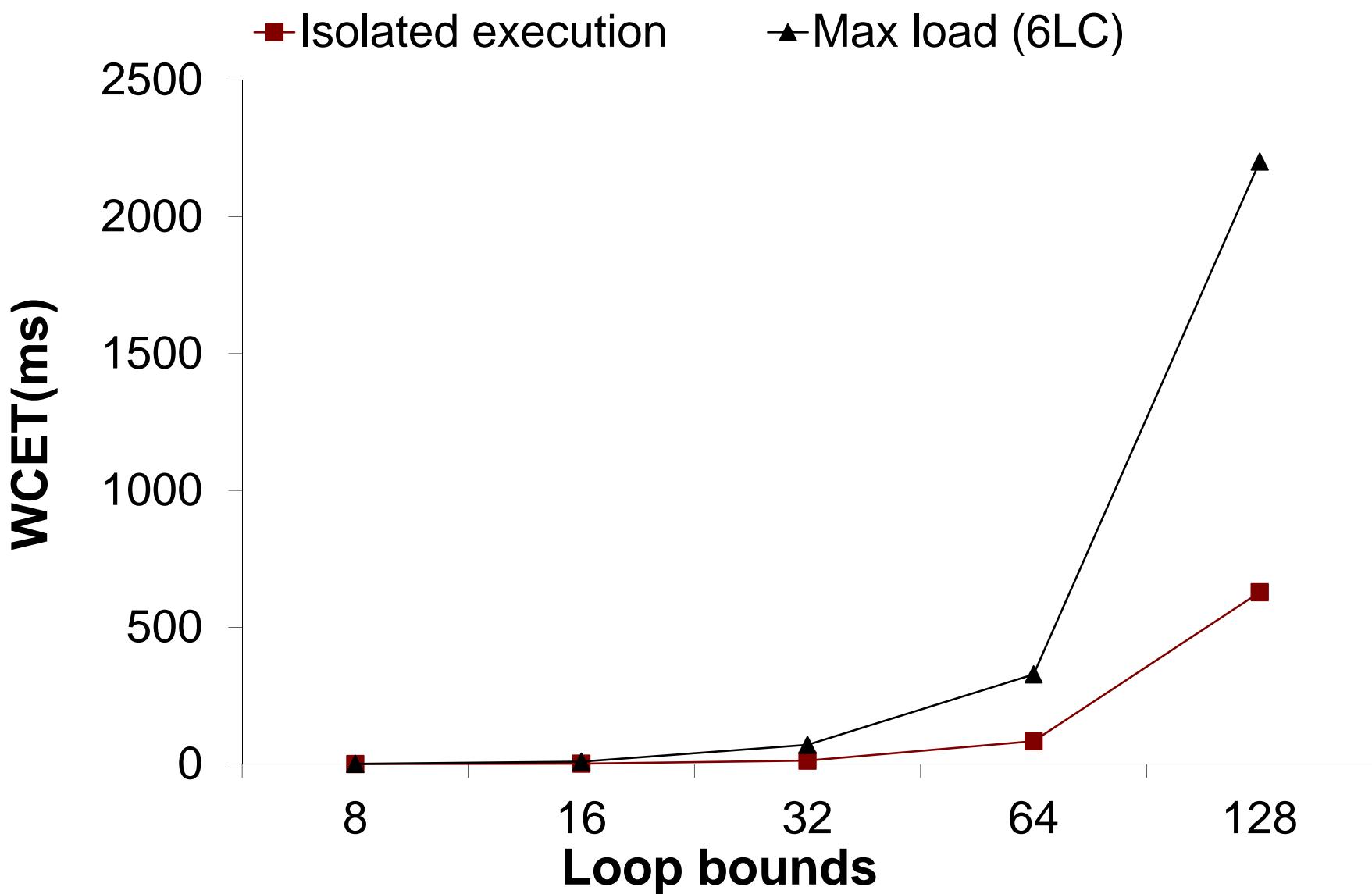
Number of Parallel tasks



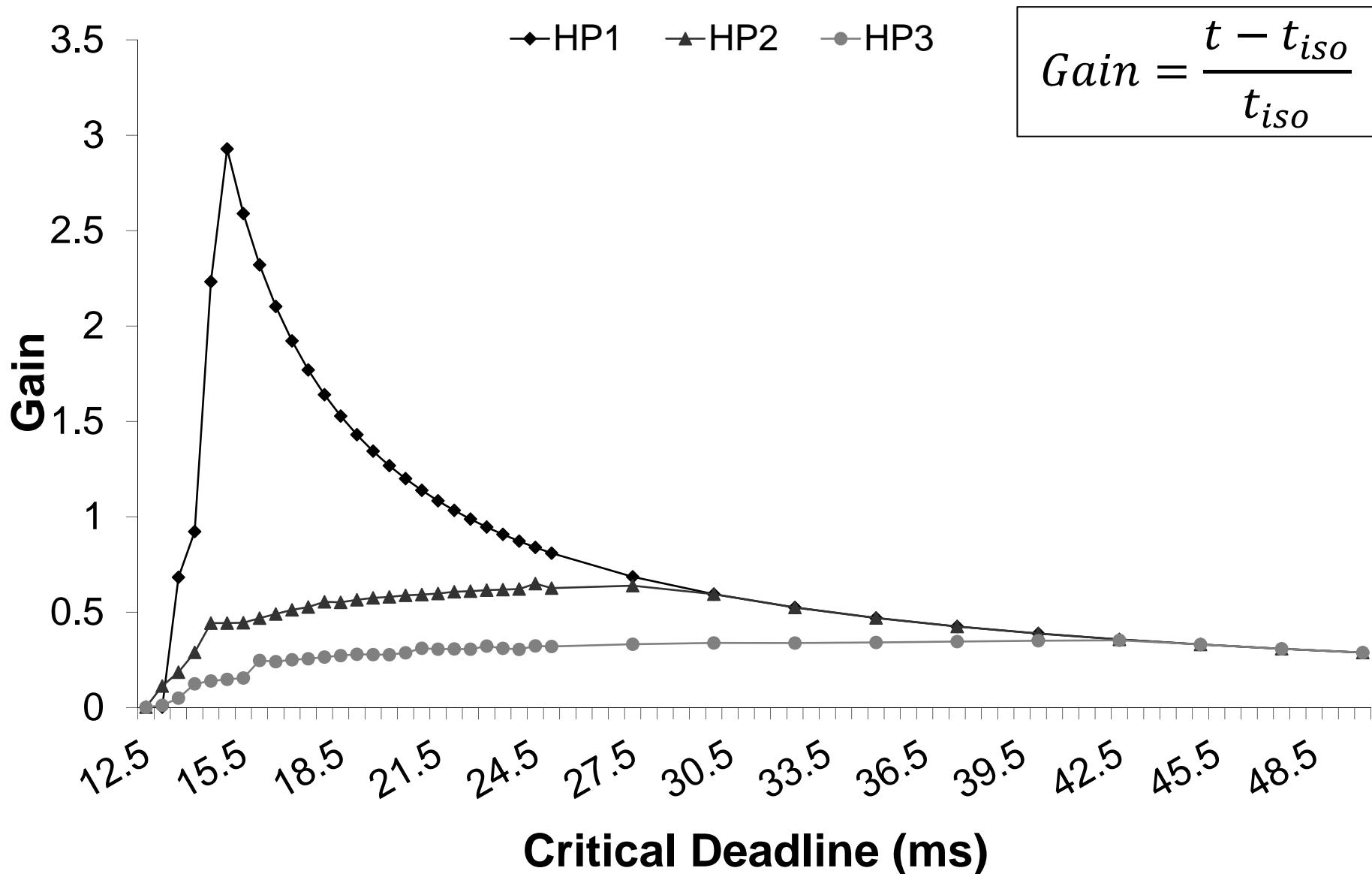
Number of Parallel tasks



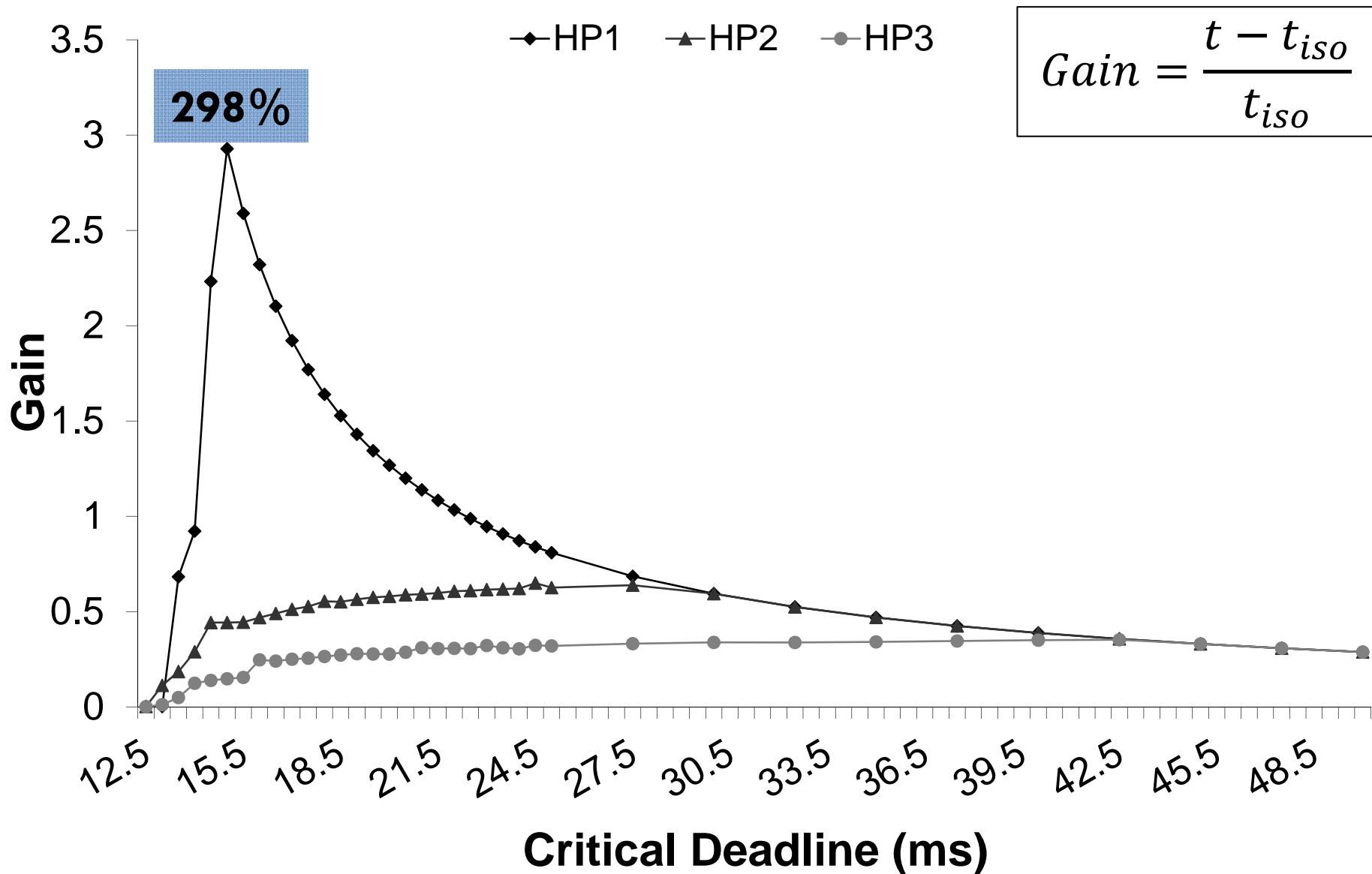
Application size



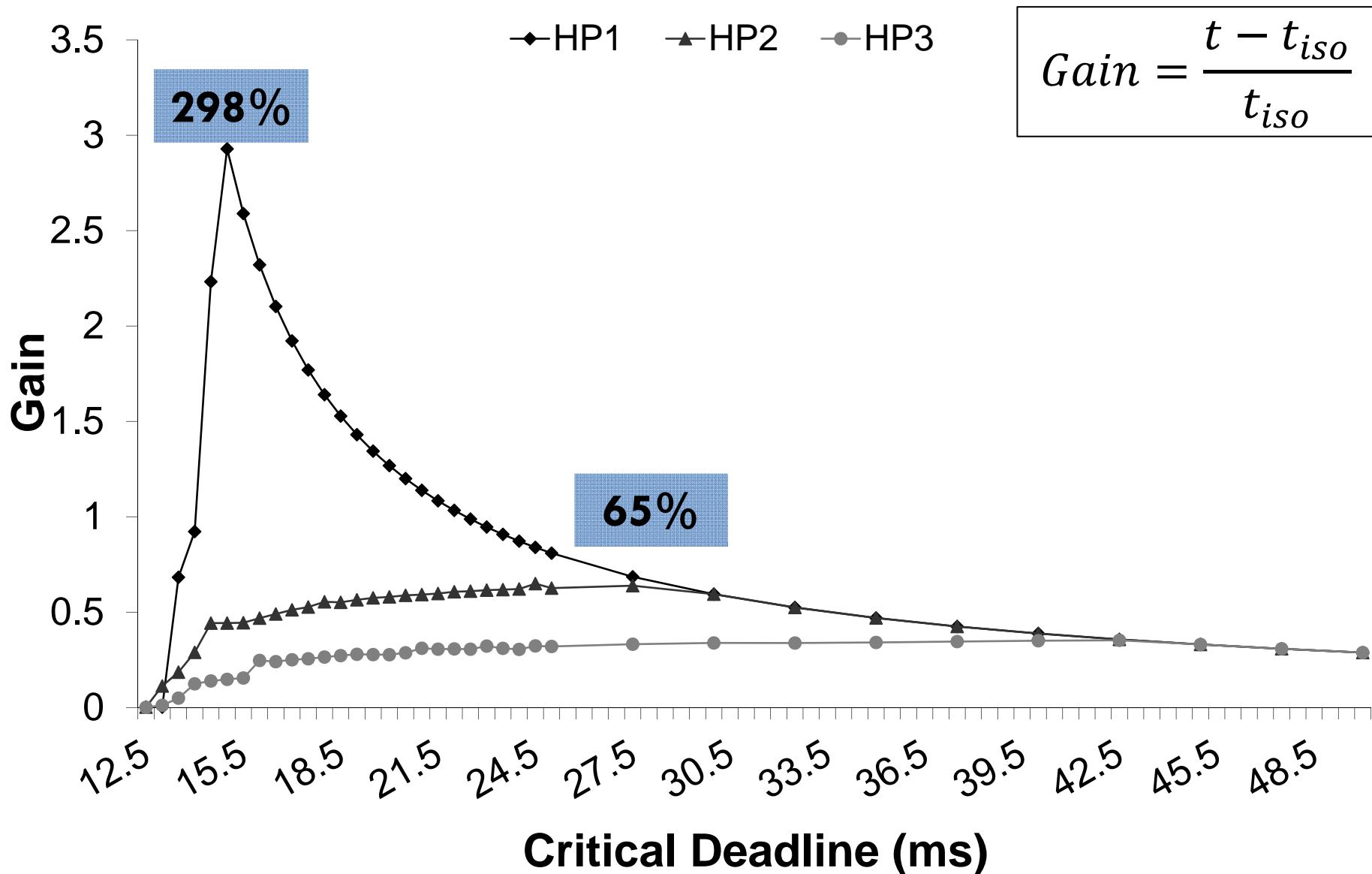
Gain (application size 32)



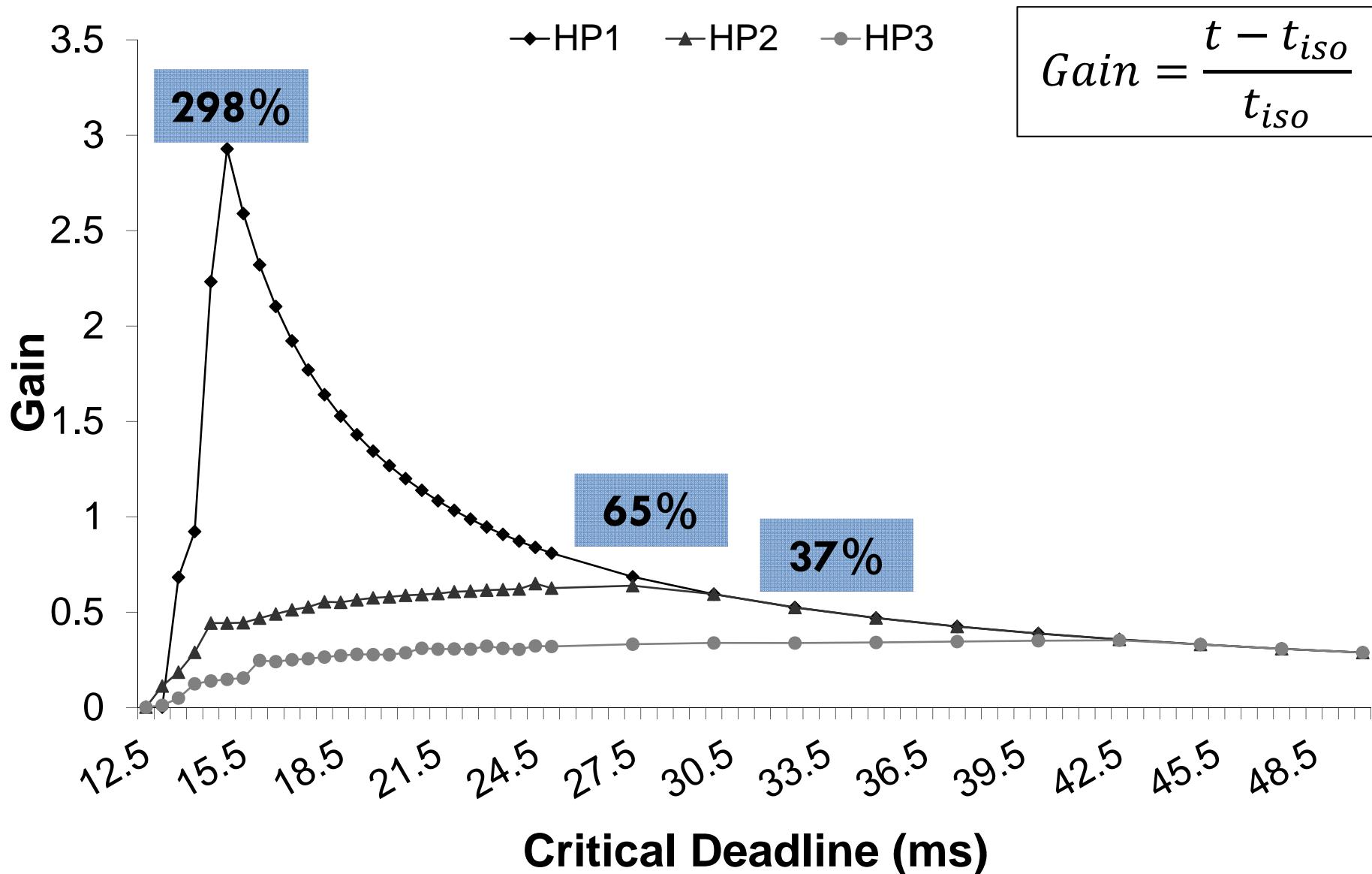
Gain (application size 32)



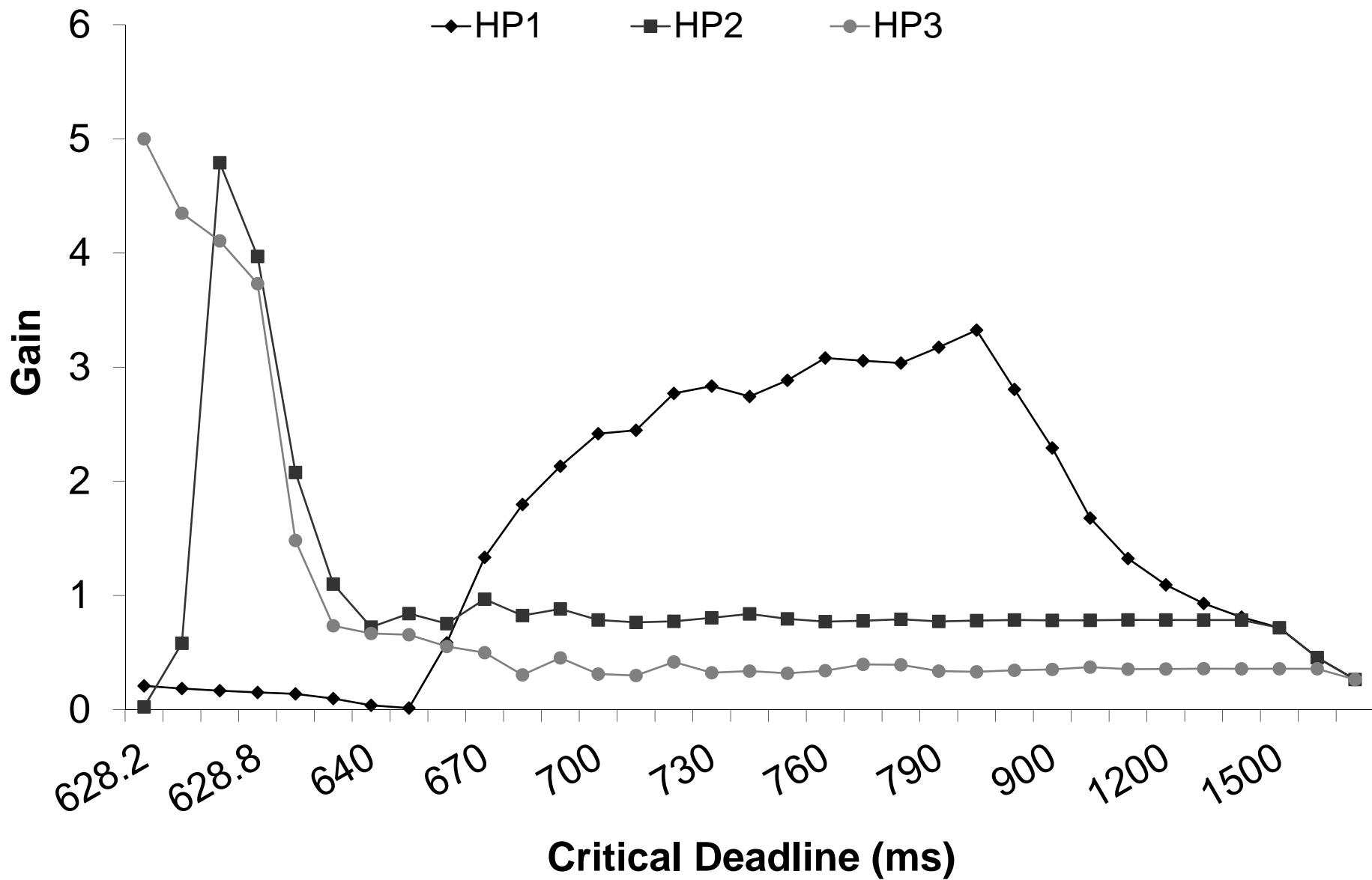
Gain (application size 32)



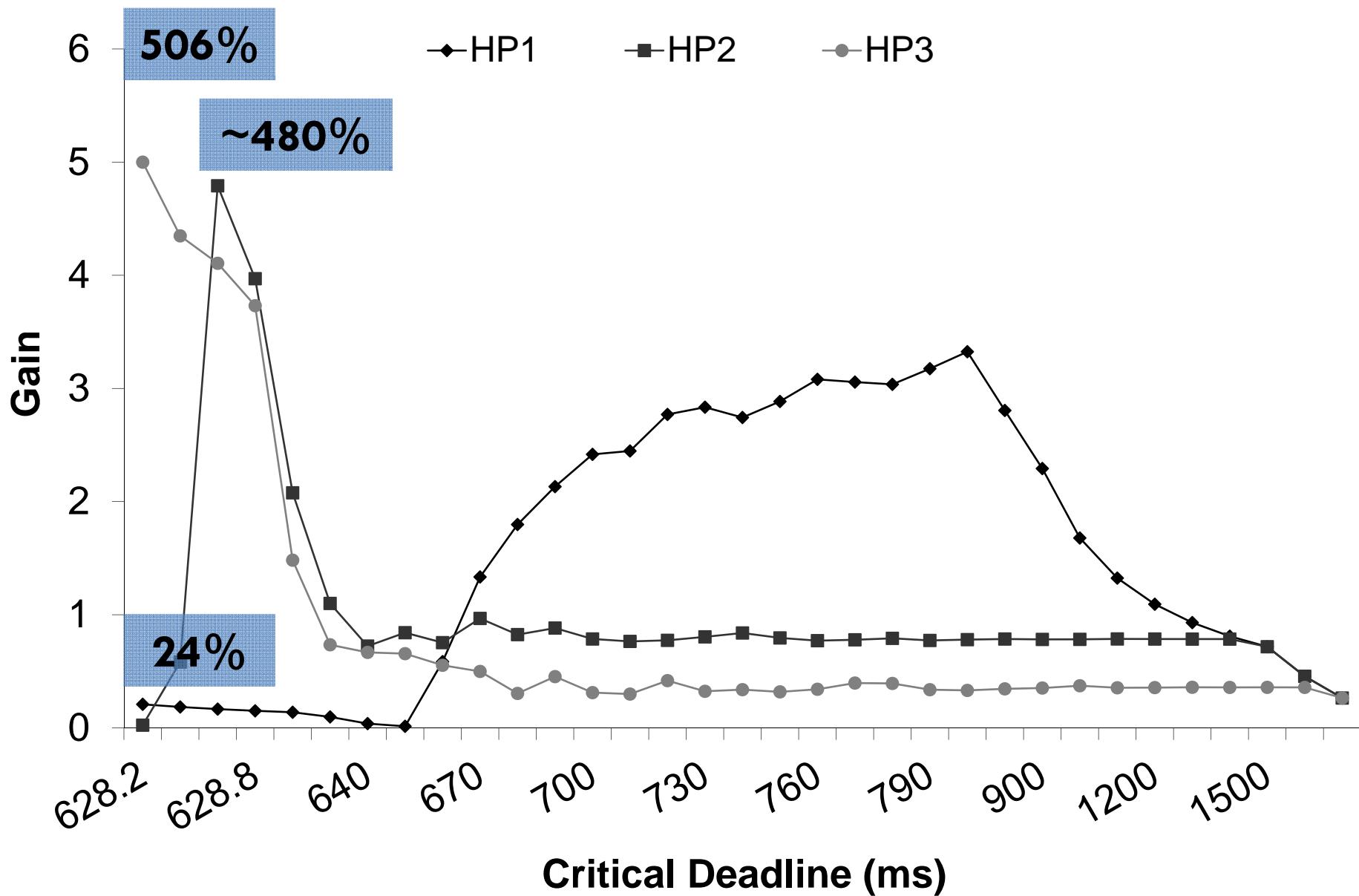
Gain (application size 32)



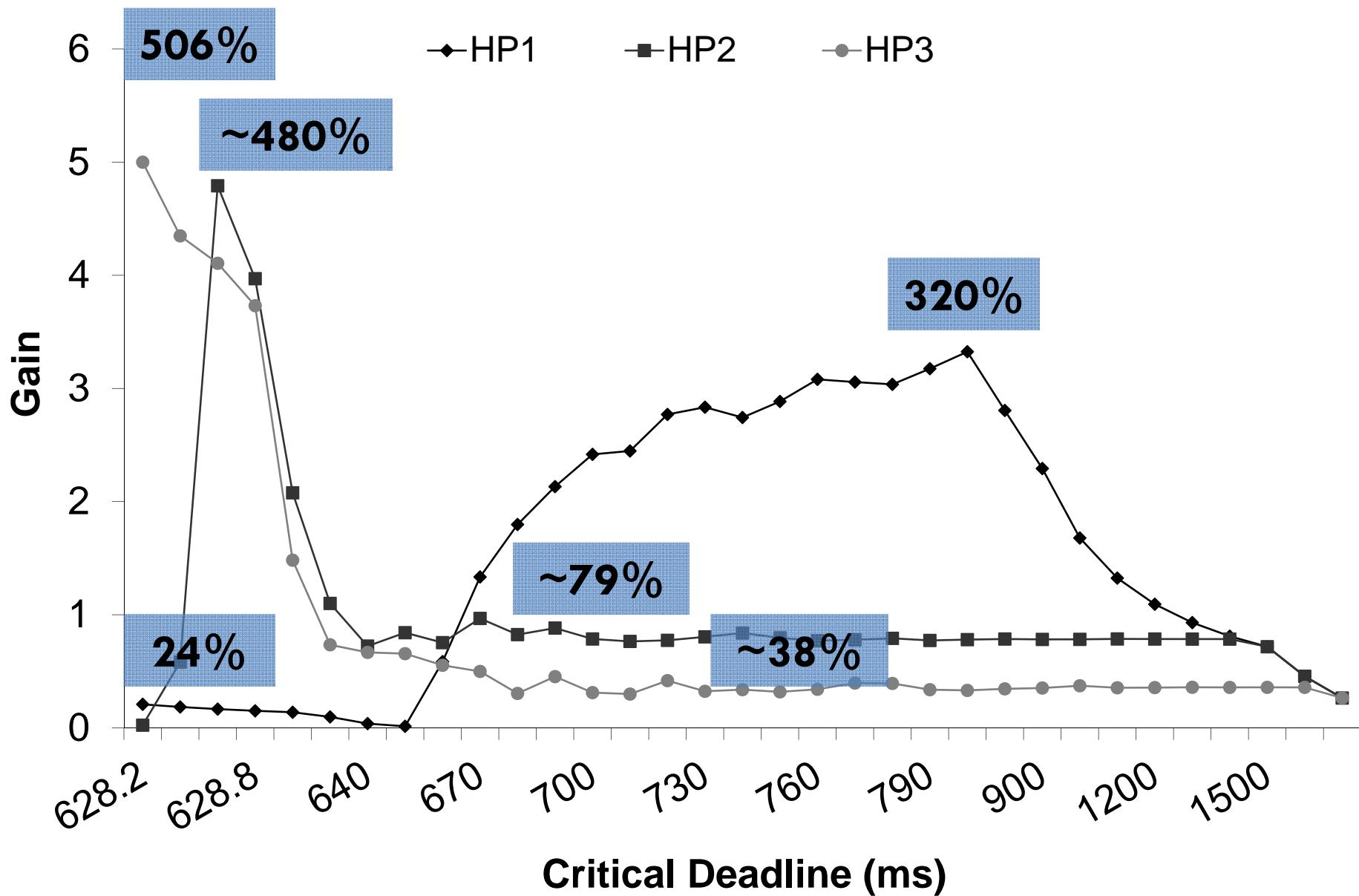
Gain (application size 128)



Gain (application size 128)



Gain (application size 128)



Conclusions & Future directions

- **Conclusions:**
 - Design-time:
 - Instrumentation of critical tasks
 - Pre-computation of structure & timing information
 - Run-time:
 - Locally use info to decide the suspension of less critical tasks
 - Globally manage the execution of the less critical tasks
 - Implementation on Texas TMS platform
- **Future directions:**
 - Methodology for position & sampling of observation points
 - Extension to:
 - Several criticality levels
 - Time & Space partitioning

THANK YOU

angeliki.kritikakou@irisa.fr

www.kritikakou.com

QUESTIONS?