Proofcraft

seL4® verification: status and plans

Michael McInerney @ Proofcraft

seL4

The world’s most highly assured operating system kernel

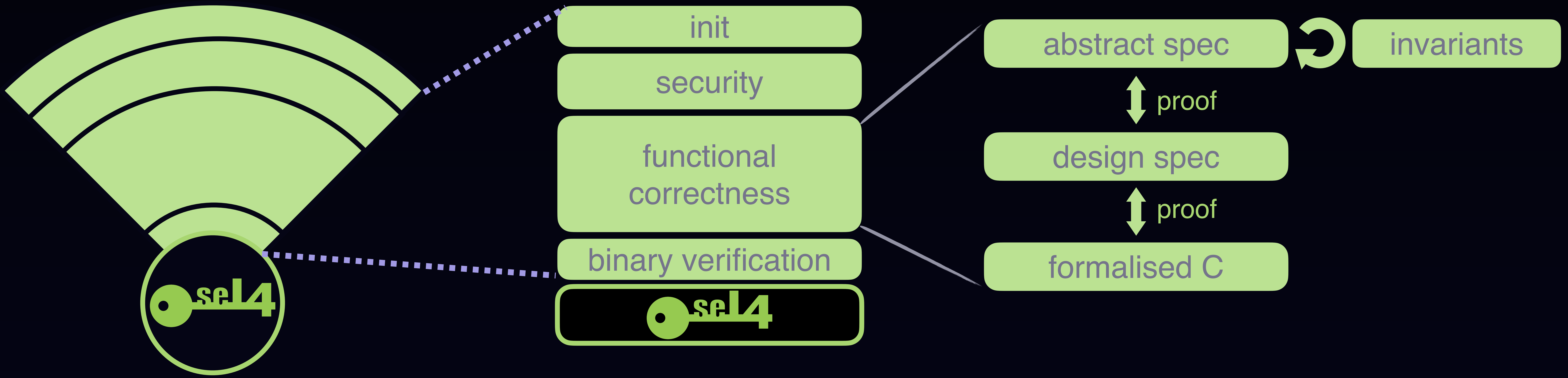
The world's most highly assured operating system kernel

Unparalleled mathematical proofs
of correctness and security

⇒ The most trustworthy foundation
for safety- and security-critical systems

Unparalleled mathematical proofs
of correctness and security

Unparalleled mathematical proofs
of correctness and security
(FC) (integrity&confidentiality)



Unparalleled mathematical proofs
of correctness and security

NOW & IN THE FUTURE

More architectures verified

More features verified

More platforms verified

More cores

Less need for
expertise & maintenance

Overview: 5 main areas Proofcraft is working on

More architectures verified	Arm 64-bit (AArch64)
More features verified	Mixed-criticality support (MCS)
More platforms verified	Automated platform port verification
Less need for expertise & maintenance	Proof architecture split (arch-split)
More cores	Verified multikernel (MK) on multicore

Overview: status & plans in a nutshell

More architectures verified	Arm 64-bit (AArch64)	
More features verified	Mixed-criticality support (MCS)	
More platforms verified	Automated platform port verification	Now: 18/32 configs verified (56%) Aim: 90-100% for existing ones + automation for new ones
Less need for expertise & maintenance	Proof architecture split (arch-split)	Done: abstract invariants Now: refinement proofs
More cores	Verified multikernel (MK) on multicore	See next talk

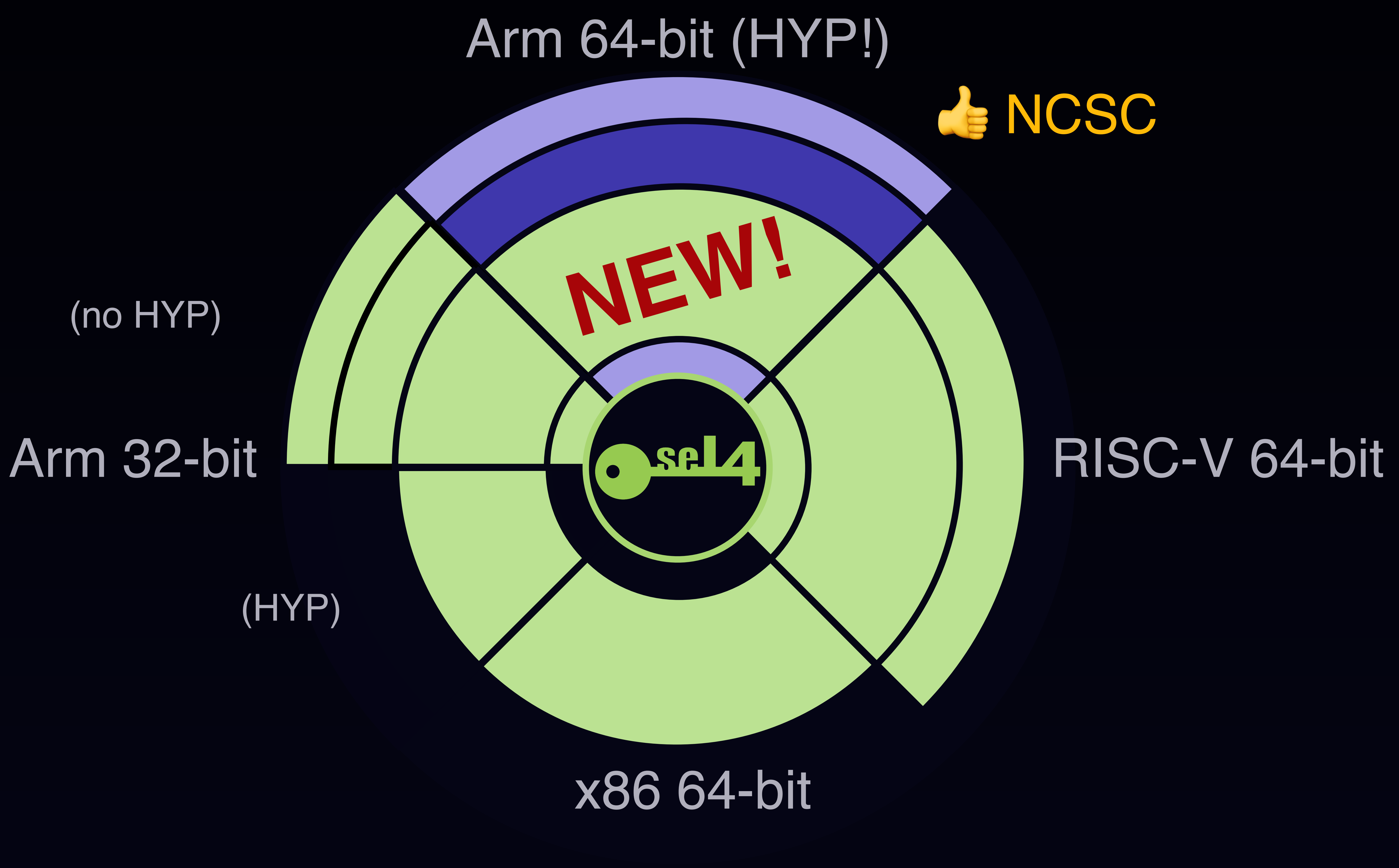
Overview: status & plans in a nutshell

More architectures verified	Arm 64-bit (AArch64)	
More features verified	Mixed-criticality support (MCS)	
More platforms verified	Automated platform port verification	Now: 18/32 configs verified (56%) Aim: 90-100% for existing ones + automation for new ones
Less need for expertise & maintenance	Proof architecture split (arch-split)	Done: abstract invariants Now: refinement proofs
More cores	Verified multikernel (MK) on multicore	See next talk

Overview: status & plans in a nutshell

More architectures verified	Arm 64-bit (AArch64)	
More features verified	Mixed-criticality support (MCS)	
More platforms verified	Automated platform port verification	Now: 18/32 configs verified (56%) Aim: 90-100% for existing ones + automation for new ones
Less need for expertise & maintenance	Proof architecture split (arch-split)	Done: abstract invariants Now: refinement proofs
More cores	Verified multikernel (MK) on multicore	See next talk

AArch64



seL4 proofs

Done
Ongoing
Future

(non-MCS, unicore)

Done: FC
Now: integrity (Q1'25)

Overview: status & plans in a nutshell

More architectures verified	Arm 64-bit (AArch64)	
More features verified	Mixed-criticality support (MCS)	
More platforms verified	Automated platform port verification	Now: 18/32 configs verified (56%) Aim: 90-100% for existing ones + automation for new ones
Less need for expertise & maintenance	Proof architecture split (arch-split)	Done: abstract invariants Now: refinement proofs
More cores	Verified multikernel (MK) on multicore	See next talk

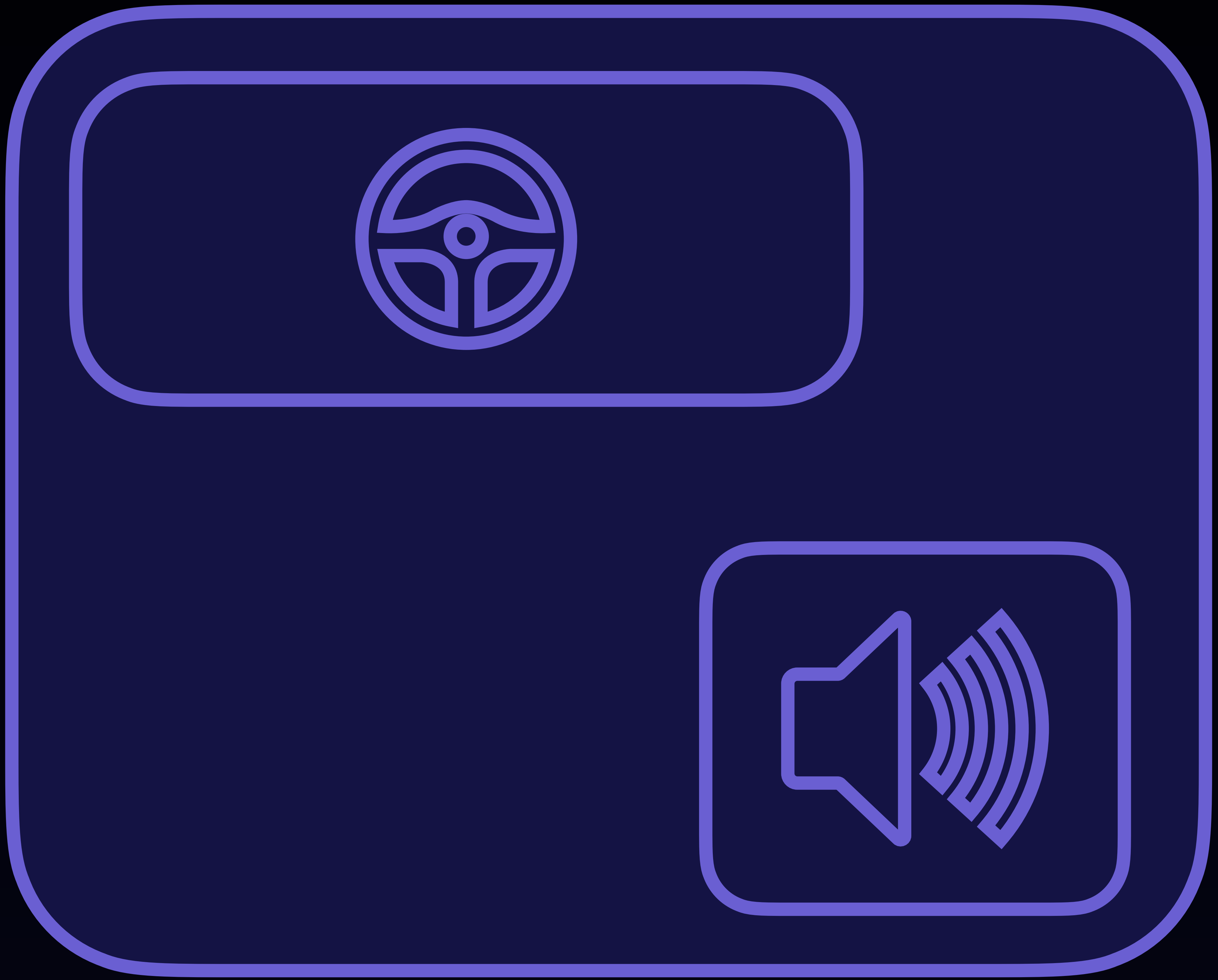
What is MCS?

What is MCS?

- Support for Mixed-Criticality Systems

What is MCS?

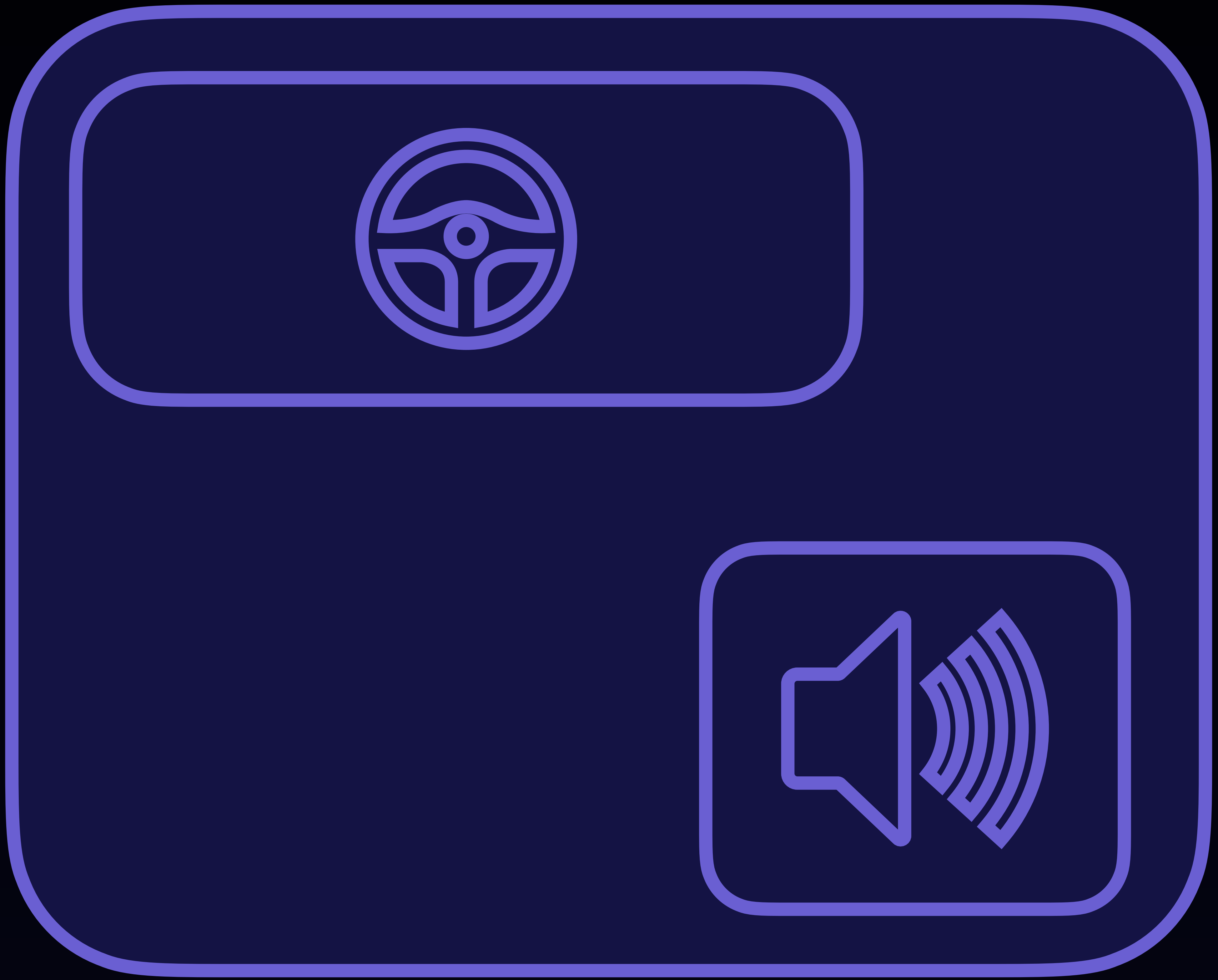
- Support for Mixed-Criticality Systems



What is MCS?

- Support for Mixed-Criticality Systems

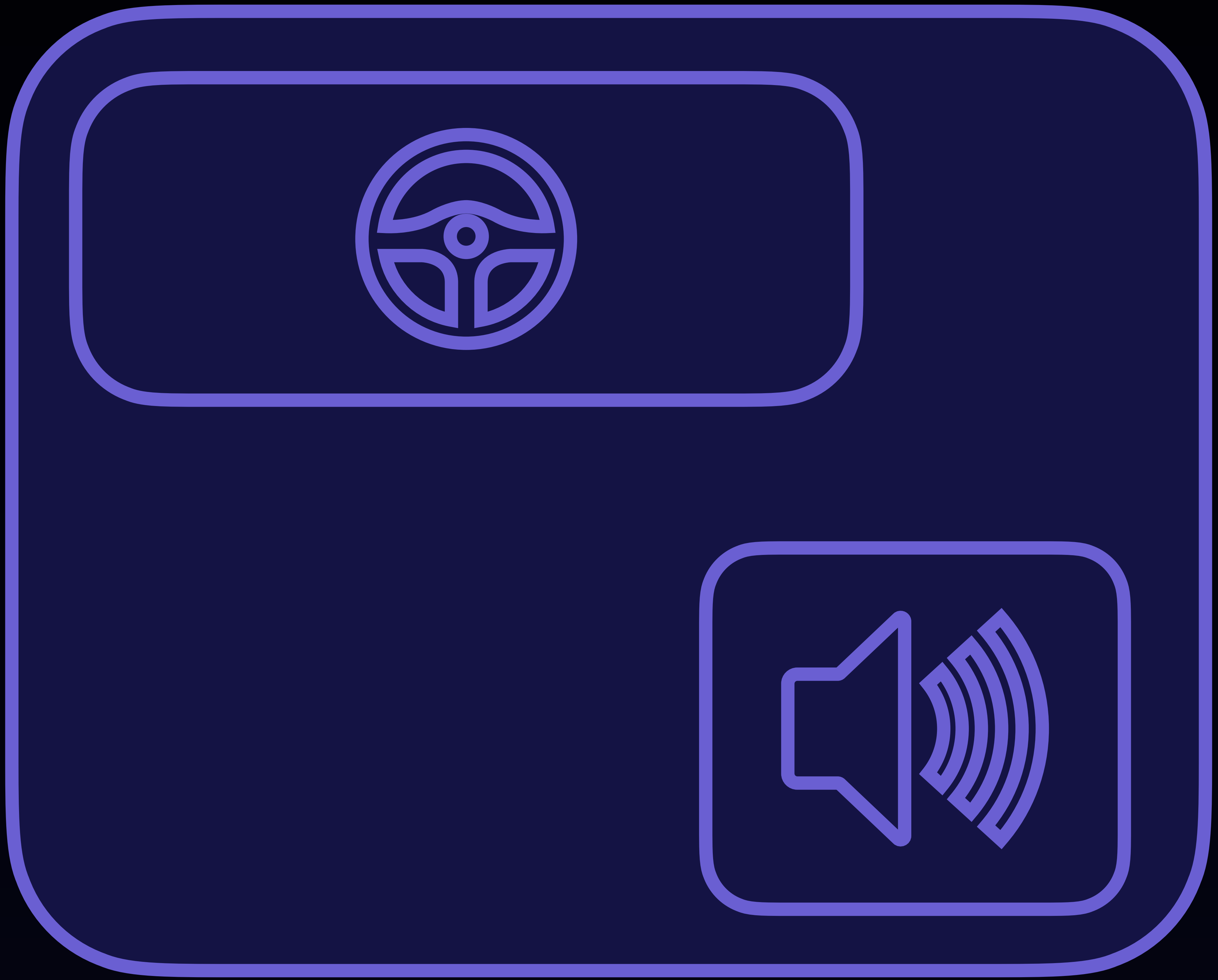
- Time as a resource



What is MCS?

- Support for Mixed-Criticality Systems

- Time as a resource
    - scheduling context objects



The proofs have evolved with new features over the years

The proofs have evolved with new features over the years

Two examples:

- bound notification endpoints
- bitfield scheduler optimisation

The proofs have evolved with new features over the years

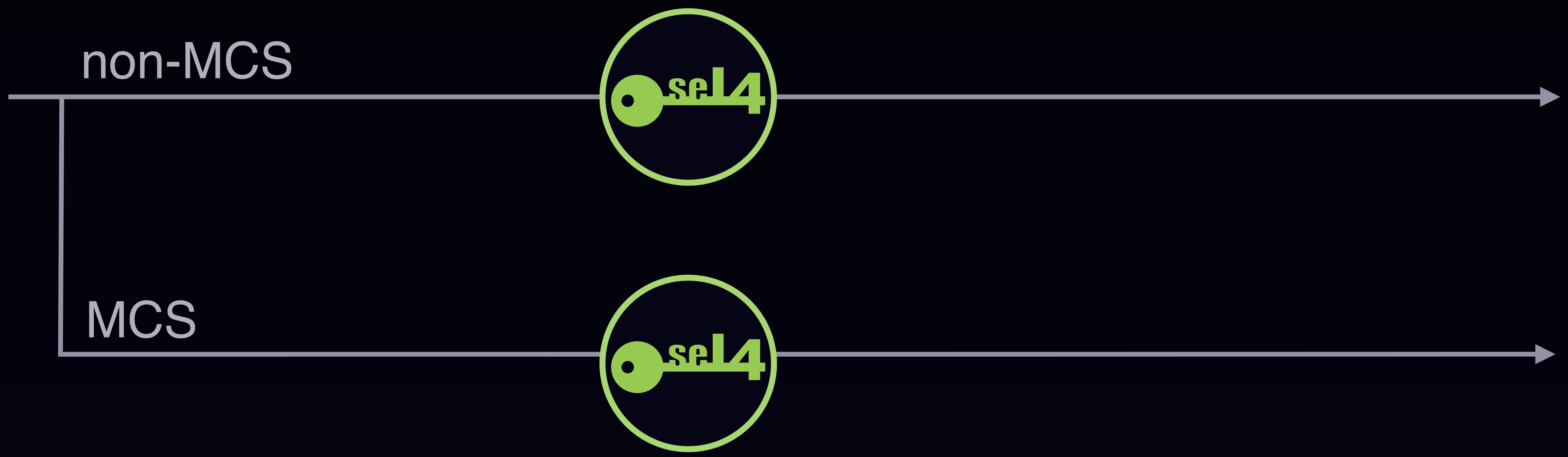
Two examples:

- bound notification endpoints
- bitfield scheduler optimisation

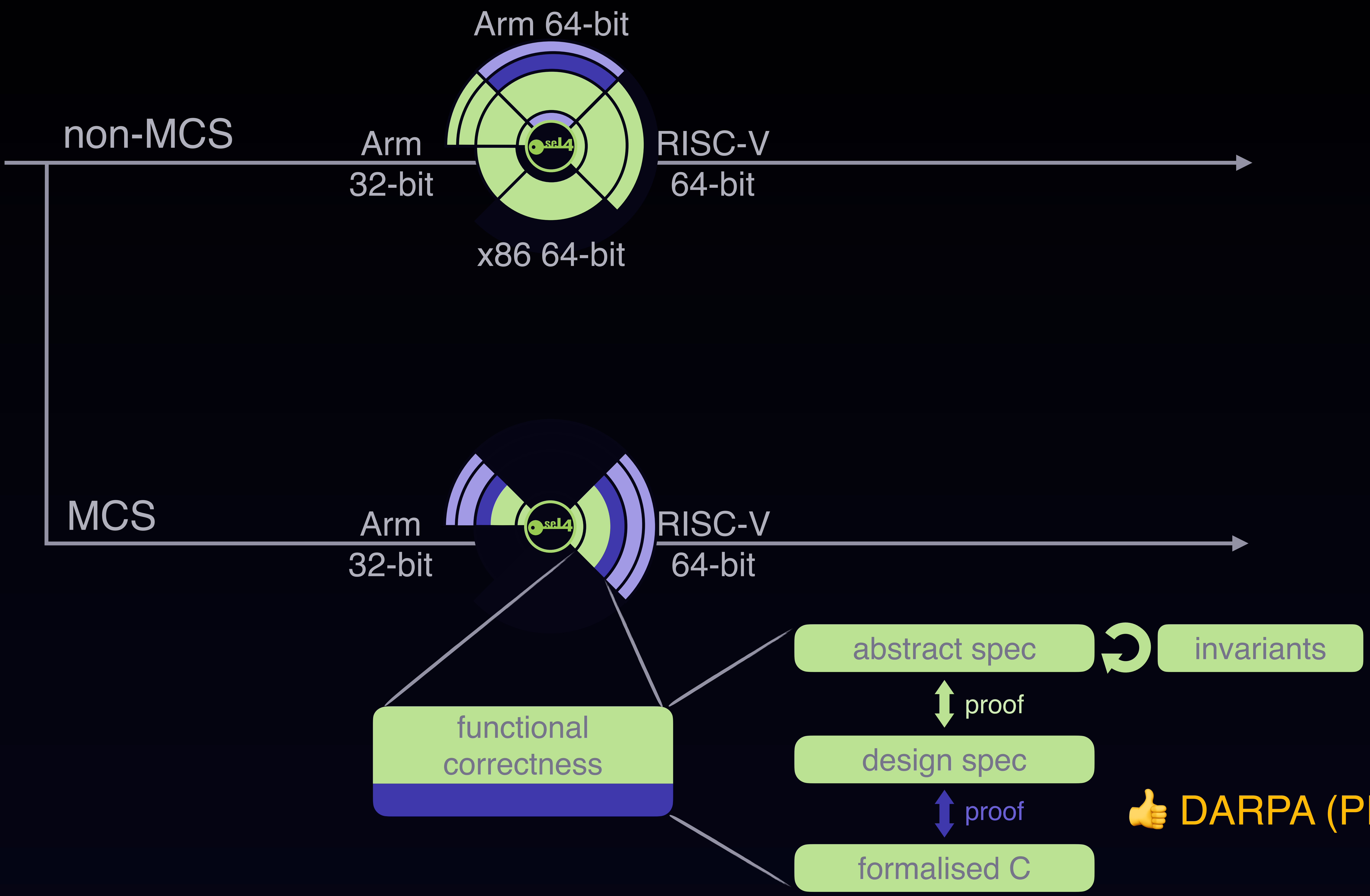
MCS is different:

- large, invasive change

Big Feature: Mixed-Criticality Systems



Verification of multiple configs in parallel



👍 DARPA (PROVERS)

What makes MCS so difficult to verify?

What makes MCS so difficult to verify?

- Many new lines of code (an ~15% increase)

What makes MCS so difficult to verify?

- Many new lines of code (an ~15% increase)
- New kernel objects (scheduling context objects and reply objects)

What makes MCS so difficult to verify?

- Many new lines of code (an ~15% increase)
- New kernel objects (scheduling context objects and reply objects)
- New invariants required, pre-existing invariants impacted

What makes MCS so difficult to verify?

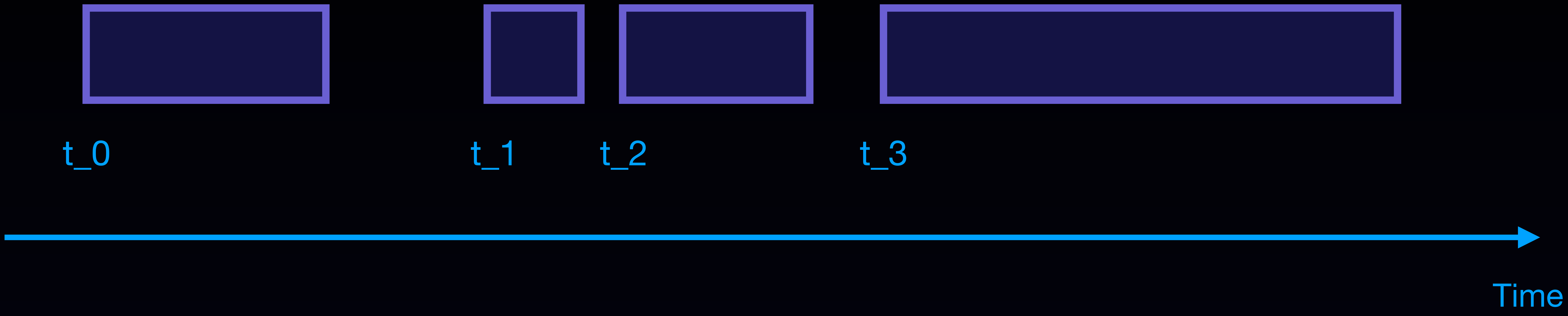
- Many new lines of code (an ~15% increase)
- New kernel objects (scheduling context objects and reply objects)
- New invariants required, pre-existing invariants impacted
- Many new functions in the kernel (in particular, new system calls)

What makes MCS so difficult to verify?

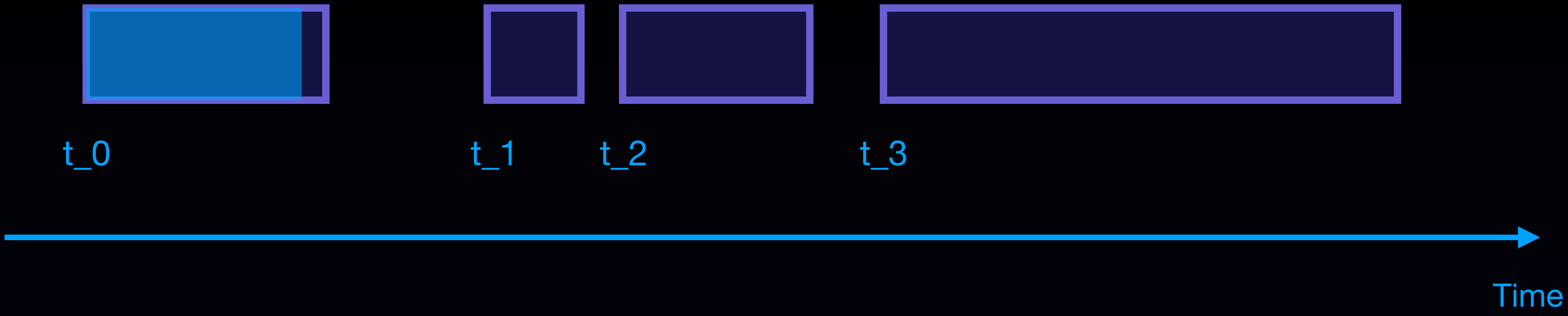
- Many new lines of code (an ~15% increase)
- New kernel objects (scheduling context objects and reply objects)
- New invariants required, pre-existing invariants impacted
- Many new functions in the kernel (in particular, new system calls)
- Many pre-existing functions are now much longer

Loops in MCS

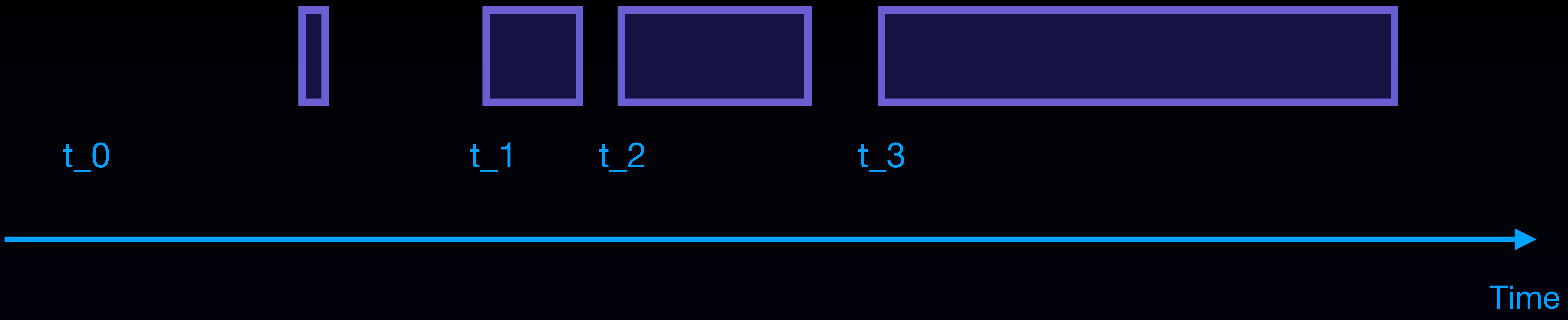
Loops in MCS



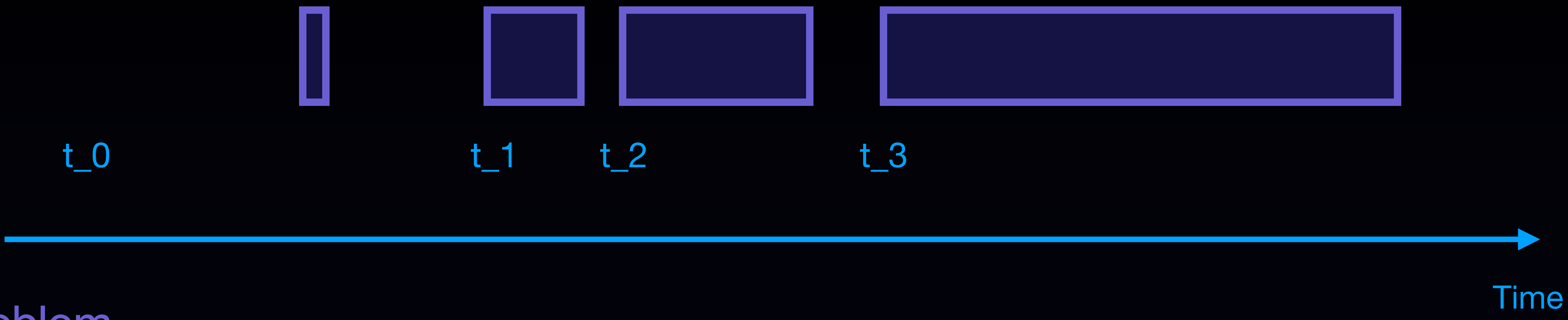
Loops in MCS



Loops in MCS

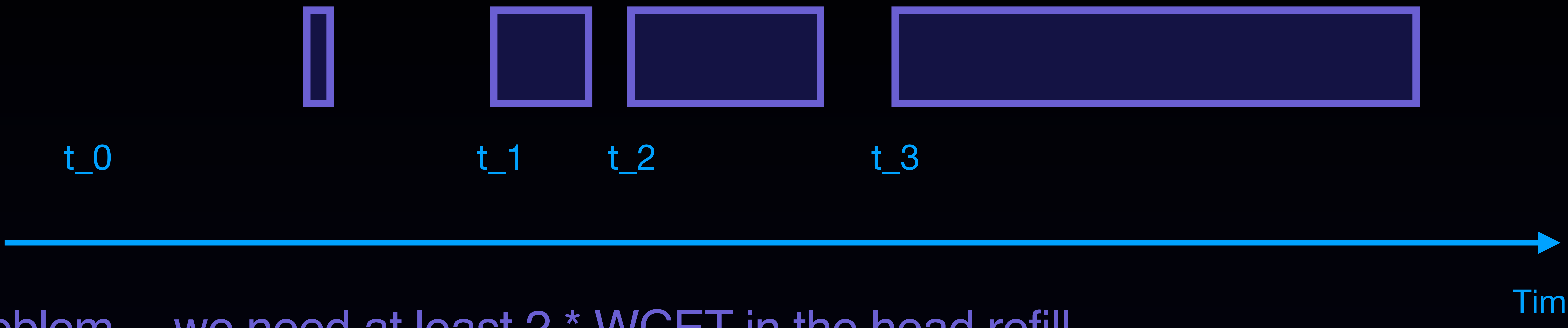


Loops in MCS



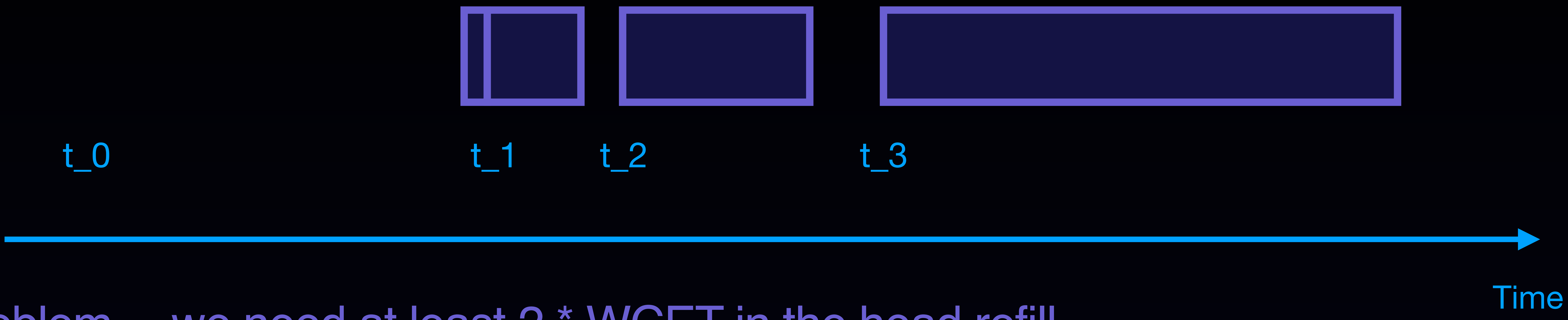
Problem…

Loops in MCS



Problem… we need at least 2 * WCET in the head refill

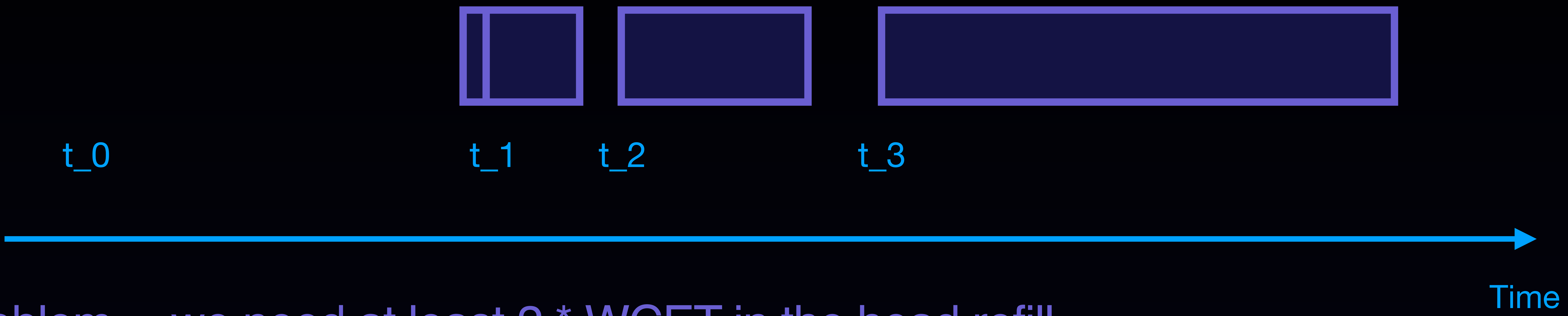
Loops in MCS



Problem… we need at least 2 * WCET in the head refill

Solution: merge refills together…

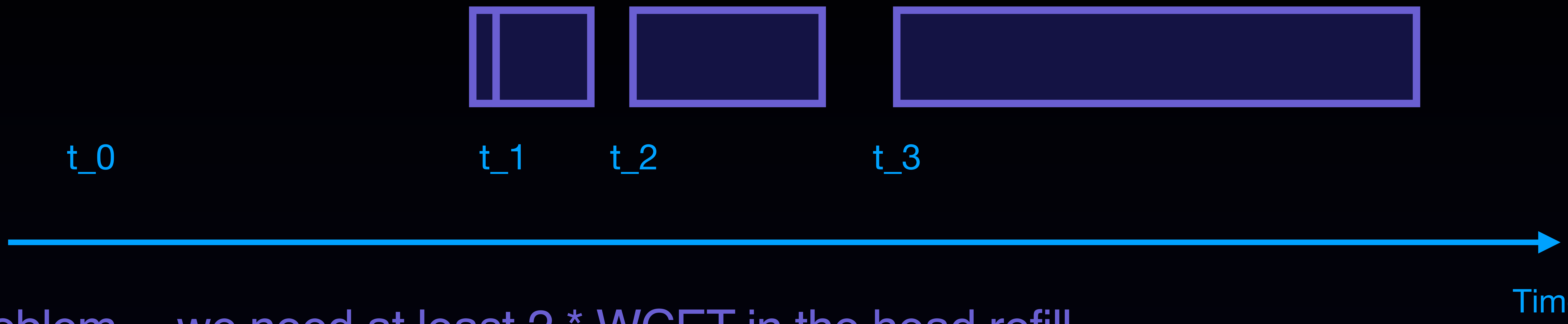
Loops in MCS



Problem… we need at least 2 * WCET in the head refill

Solution: merge refills together… and keep merging refills until we have at least 2 * WCET (overflow?)

Loops in MCS

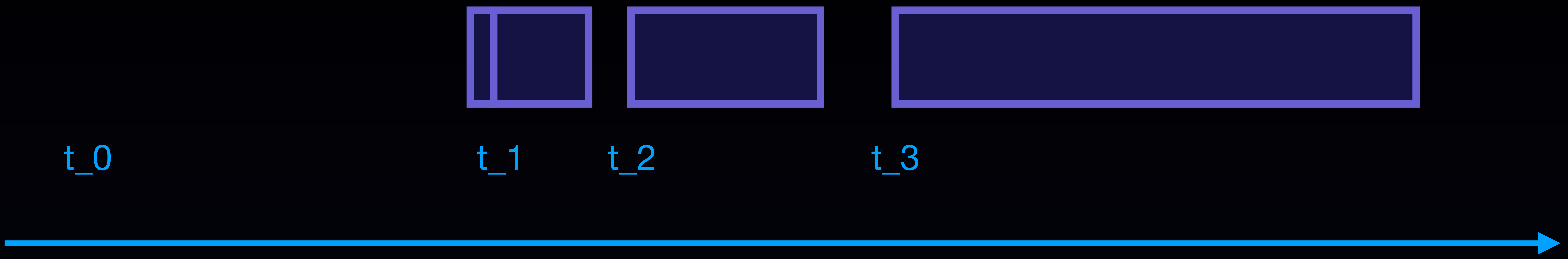


Problem… we need at least 2 * WCET in the head refill

Solution: merge refills together… and keep merging refills until we have at least 2 * WCET (overflow?)

Termination?

Loops in MCS

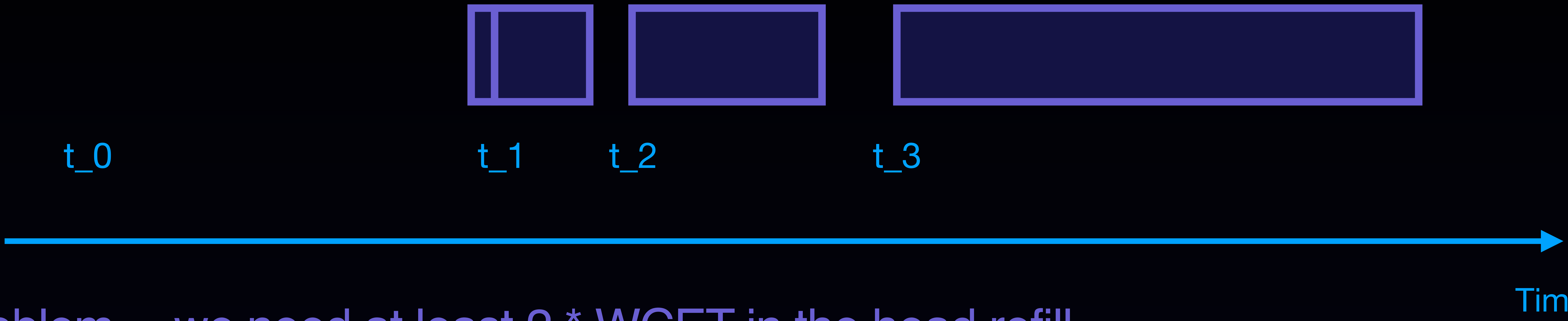


Problem… we need at least 2 * WCET in the head refill

Solution: merge refills together… and keep merging refills until we have at least 2 * WCET (overflow?)

Termination? The scheduling context must have a budget of at least 2 * WCET —> new system-wide invariant

Loops in MCS



Problem… we need at least 2 * WCET in the head refill

Solution: merge refills together… and keep merging refills until we have at least 2 * WCET (overflow?)

Termination? The scheduling context must have a budget of at least 2 * WCET —> new system-wide invariant

Data refinement? Ring buffer of refills versus list of refills

Conclusion

Unparalleled mathematical proofs of correctness and security

- More architectures verified
- More features verified
- More platforms verified
- More cores
- Less need for expertise & maintenance

https://proofcraft.systems