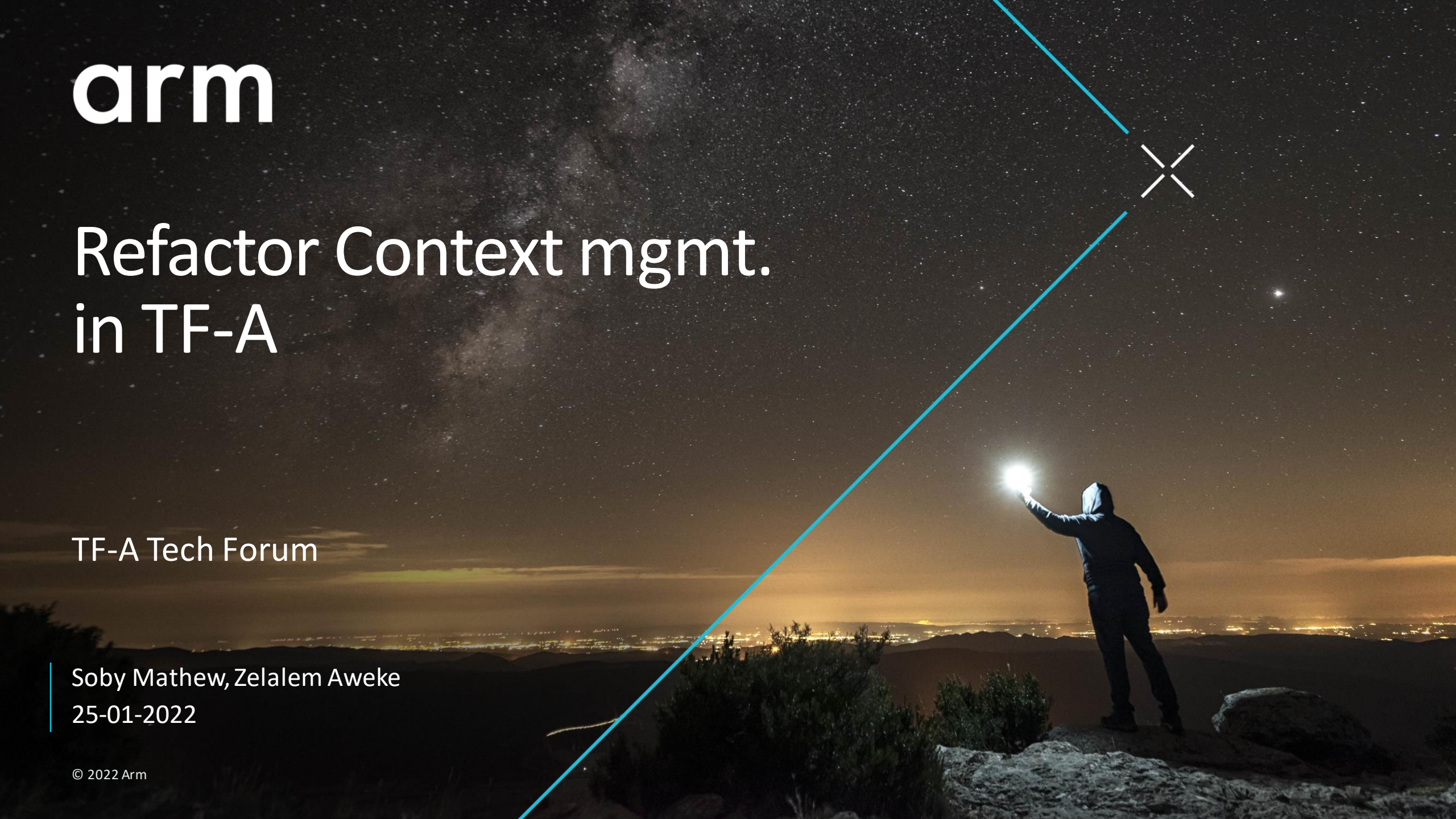


Refactor Context mgmt. in TF-A

TF-A Tech Forum

Soby Mathew, Zelalem Aweke

25-01-2022

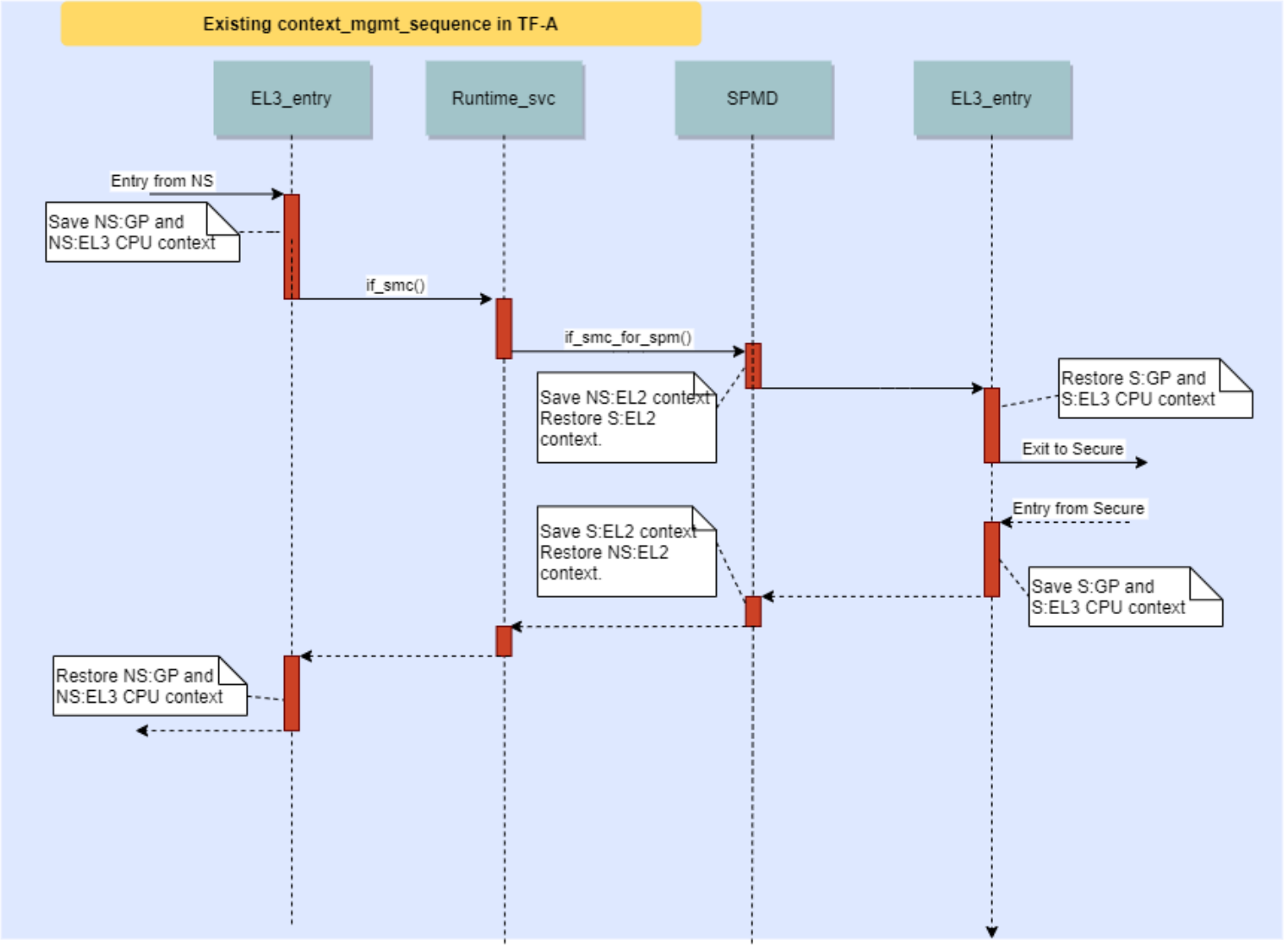
Introduction

- The context management library in TF-A provides helper routines for initialization and switching CPU context during world switch.
  - BL31 maintains CPU Context for each world which is initialized during cold boot. CPU features are enabled depending on the EL3 sysregs programmed.
  - At runtime, depending on the SMC/event, SPMD/RMMD invokes context management helpers to save and restore the context.
- Due to the gradual evolution of this library over time, the current state has made it a maintenance hazard and prone to programming errors.
- The agenda is to discuss the overall design direction on how to refactor the lib.
  - Some of the details may undergo further refinement if implementation shows difficulties.

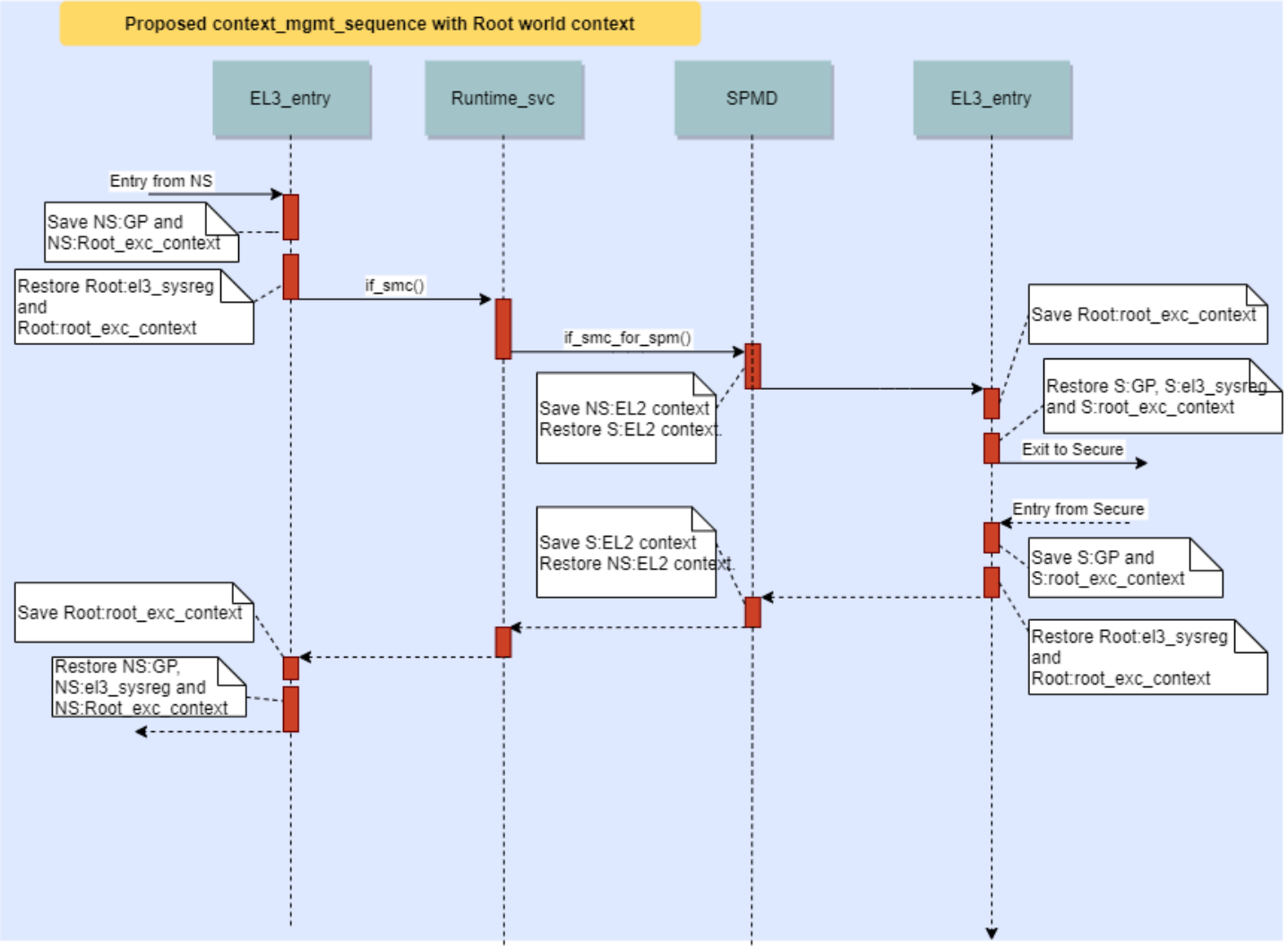
Design Principles

- **Decentralized model for context mgmt.**
  - **See cm_setup_context()**
  - **Move world responsibility to world dispatcher and decentralize the management.**
- **EL3 should only initialize immediate used lower EL**
  - **Since it is likely that EL2 is managing the EL1 context , EL3 need not initialize this.**
  - **To maintain confidentiality between worlds, it may be necessary to restore EL1 context.**
    - **Depends on how SPM and RMM is managing EL1 context.**
- **Maintain EL3 sysregs which affect lower EL within CPU context**
  - **Allows per-world control of traps/feature enablement.**
  - **This is the pattern followed more or less in code today.**
- **Allow more flexibility for Dispatchers to select feature set to save and restore**
  - **See el2_sysregs_context_save()**
  - **Break up the monolithic into several smaller functions and allow dispatchers to choose.**

Introducing the Root Context – why is it needed ?



Context mgmt. with Root context



arm

Proposed Changes

1. Cleanup context registers

Cleanup EL3 context

- Add more registers that control features/traps for lower Els to EL3 context

- Registers that don't need to have different values across the worlds can be removed from this context

Cleanup EL2 context

- Remove registers that are only accessible from Secure state from EL2 context

2. Cleanup *cm_setup_context*

- Split *cm_setup_context* and move security state specific logic to corresponding functions:
  - *cm_setup_common* – common inits
  - *cm_setup_secure_context* – Secure state specific inits, used by SPMD
  - *cm_setup_realm_context* – Realm state specific inits, used by RMMD
  - *cm_setup_ns_context* – NS state specific inits, used by PSCI/BL31
- Same interface as *cm_setup_context*
- Legacy code can keep using *cm_setup_context* and can transition gradually

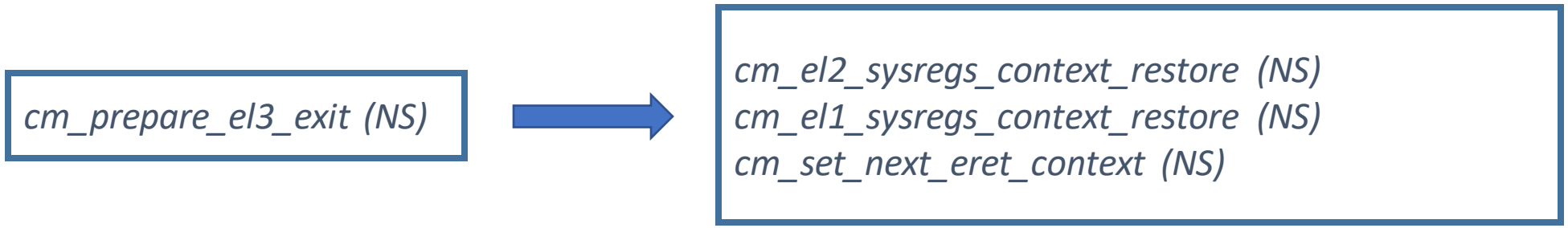
3. Alternative flow for first exit to NS world

- Today *cm_prepare_el3_exit* is used to exit to NS world from EL3 the first time

- Current implementation does the following:
  1) Directly initializes EL2 registers for two NS cases:
     - For exiting to NS-EL2 (HYP mode)
     - For exiting to NS-EL1 (SVC mode) - skip EL2 config
  2) Enables features for non-secure case
  3) Restores EL1 context with *cm_el1_sysregs_context_restore* for all cases (secure/non-secure)
  4) Sets next context with *cm_set_next_eret_context*

Alternative flow when CTX_INCLUDE_EL2_REGS is enabled

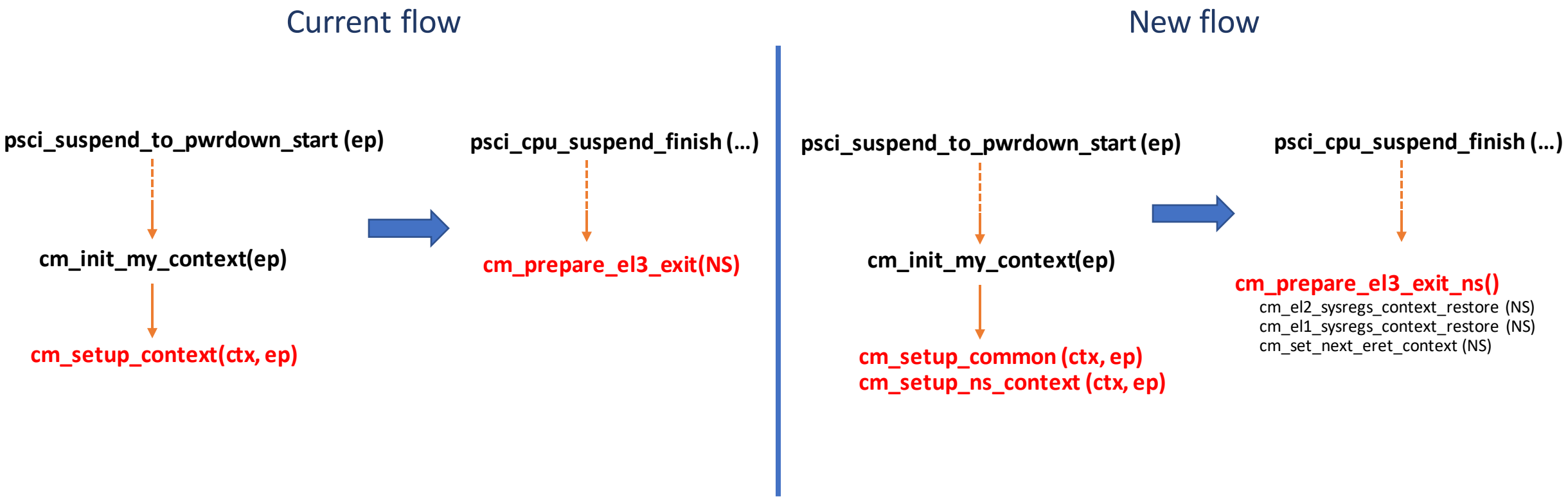
- Move (1) and (2) to *cm_setup_ns_context* and do the initializations **using context registers**
- Then later restore NS context, similar to what SPMD and RMMD do:



**cm_el1_sysregs_context_restore (NS) is needed for exit to EL2 to clear any latent values in EL1 regs after Secure world initialization. But this can likely be optimized.*

- If legacy cannot be supported after rework, a variant *`cm_prepare_el3_exit_ns` will be created.*

e.g. PSCI CPU power down



4. Cleanup *cm_el2_sysregs_context_save/restore*

- Currently these functions call *el2_sysregs_context_save/restore* assembly functions which do all EL2 register saving/restoring

- New proposal:
  - Use *el2_sysregs_context_save/restore* only for common registers (maybe rename to *el2_sysregs_context_save/restore_common*)
  - Move feature specific register save/restore out of *el2_sysregs_context_save/restore* into their own assembly functions eg : *el2_sysregs_context_feat_xxx_save/restore*
  - *SPMD and RMMD* can choose which functions to call dynamically or using feature build flags

5. Add Root context

- Prototype the changes and assess the impact on the code base
  - Identify registers that should be part of *root_exc_context*
  - Implement prototype to assess the impact on binary size and SMC call response latency
- Depending on results, we need to see how best to take it forward

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה

Click to add text