Linaro Connect
MADRID 2024 | MAY 14-17 2024

# Recent implementation/refactoring in TF-A

Manish Pandey, Arm

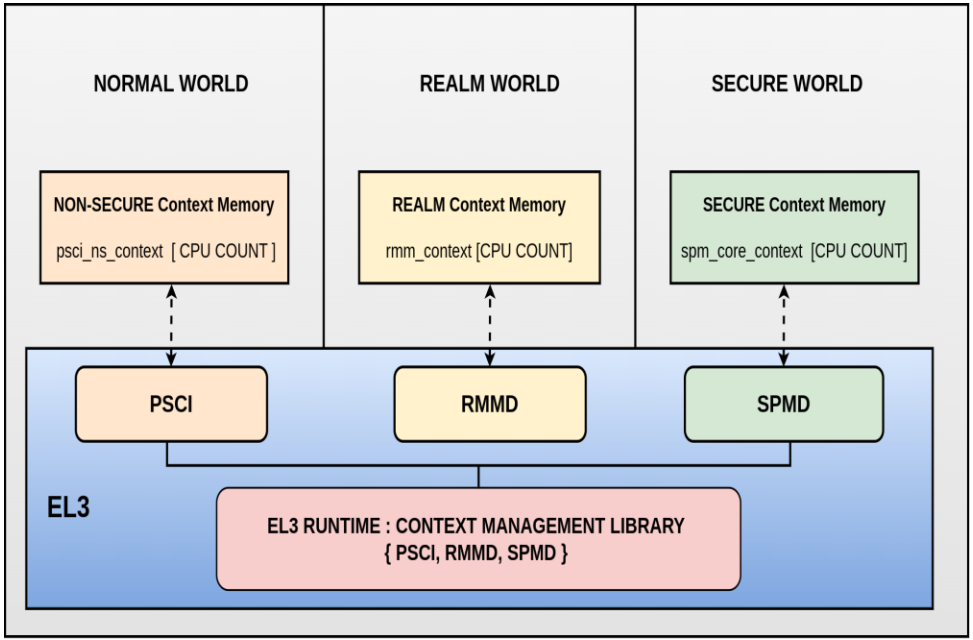
# Content

- Context management
- Exception handling
- Undefined Injection support
- Firmware Handoff
- Vendor specific EL3 monitor SMC
- Feature Detection mechanism
- Enabling Architecture Features

# Context Managment

- TF-A (BL31) provides CPU context initialization and world switching routines
  - Original design was based on 2 world system (EL3 being secure)
  - Arm CCA introduces 2 new worlds Realm and Root
  - Need refactoring to scale for 4 world system and decouple EL3 own context from secure
- Design principles [1] for 4 world system
  - Reduce EL3 FW complexity and footprint
  - EL3 should initialize immediate used lower EL
  - Decentralized model
    - The dispatchers (in EL3) should save/restore EL2 regs (SPMD/RMMD)
    - EL1 system registers should be saved/restored by EL2 managers (SPM/RMM)
  - Flexibility to pick and choose feature registers to be saved/restored

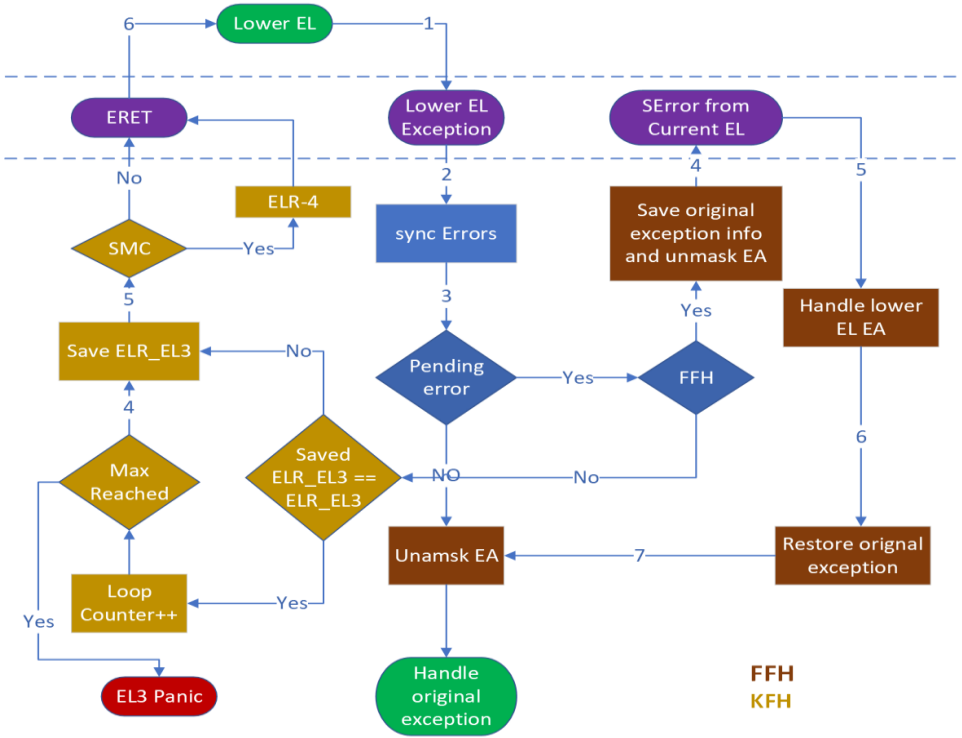
- Major changes
  - Migrate EL2/EL1 context from assembly to C
  - Allocate context memory based on feature enablement
  - Restrict few EL3 registers to have per-world instead of per CPU copy
  - Introduce root context
  - Utility to report context memory usage



# Exception Handling

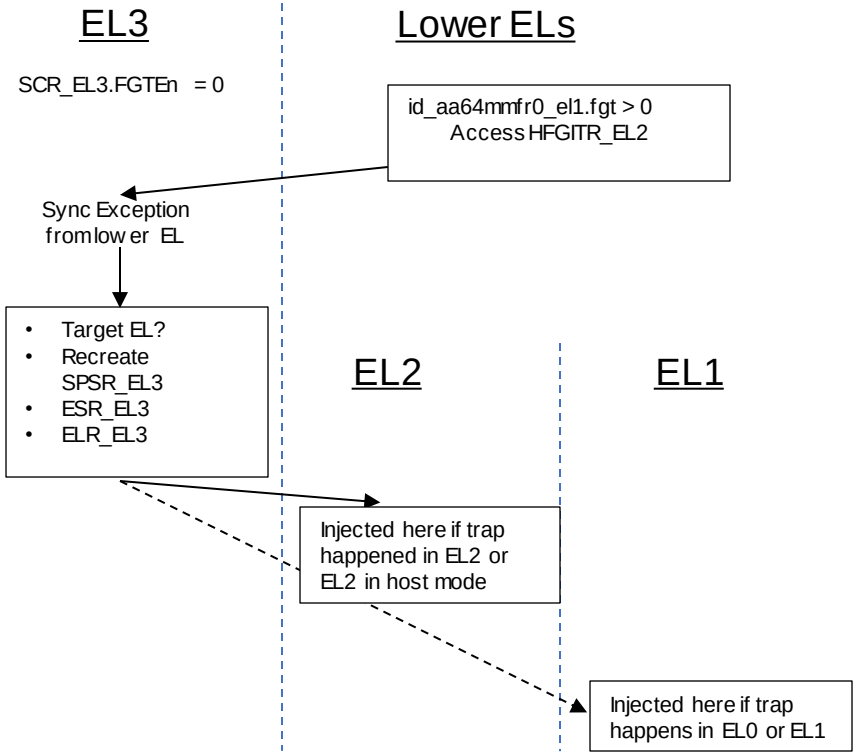
- Error synchronization at EL3 vector entry paths
  - System error which happened in lower EL but not yet signalled to PE (rare)
  - Enter EL3 because of lower EL exception (e.g. SMC)
  - The error will be signalled as EL3 async EA when EA is unmasked during EL3 execution
- On detecting pending async EA during EL3 entry, based on EA routing model
  - Firmware First handling (FFH)
    - Handle the pended EA first and then handle original exception
  - Kernel First handling (KFH)
    - Reflect error back to lower EL without handling original exception
- Behaviour with FEAT_RAS [2]
  - Avoid "esb" instruction as it might consume the error
  - TF-A relies on FEAT_IESB (assumed to be present if RAS is present)

# Error Synchronization at EL3 entry



# Undefined injection

- Lower ELs use ID registers to get PE capabilities while EL3 mostly enables them statically
- For a given arch feature, trap/enable typically combined in same control bit of EL3 register
  - Previously RES0 bits used for new feature control
  - Old EL3 FW unaware of this feature programs it to 0 (different semantics)
  - If accessing feature system register traps to EL3, it can end up in EL3 panic
  - The problem is, a VM can cause EL3 panic
  - Safety net : Inject this error back to lower EL, it may take some actions (e.g. avoid using the feature or update the FW)



# Firmware handoff

- Why?
  - Standardize the data structures and register convention between boot stages
  - Universal, Architecture agnostic and lightweight data structures
- What?
  - Transfer list (TL) comprised of various Transfer entries(TE) to propagate information during boot phases
  - Each TE represents the information produced by a boot stage and intended to be consumed by any later stage(s)
- Where?
  - Firmware handoff specification(0.9) is evolving and plans for 1.0 version soon [3]
  - Initial support in TF-A(BL1, BL2, BL31), OP-TEE and U-boot
    - U-boot's existing bloblist is identical in concept, update to make it fully compatible by adding TL attributes and extending the APIs

# Firmware Handoff

- Future
  - Currently each project has its own copy of library, what about hosting it as standalone library along with a tool (like libfdt and dtc)?
  - FVP support is experimental now, make it default choice
  - Wider adoption by other FW projects?

| TE type | Range | Description | Examples |
|---|---|---|---|
| Standardized TE | 0 ~ 0xff | Components that are very commonly used cross multiple projects | FDT, HOB, ACPI |
| | 0x100 ~ 0x1ff | Trusted Firmware related projects | OPTEE pageable part, SPMC manifest, TB_FW_CONFIG, EP_INFO |
| | 0x200 ~ 0x7f_ffff | Other standardized components | |
| Reserved | 0x80_0000 ~ 0xff_efff | Reserved for future extensions of Standardized TE | |
| Non-standardized TE | 0xff_f000 ~ 0xff_ffff | Private area for single projects | |

# Vendor specific EL3 monitor SMC

- Why?
  - Platform independent service from an EL3 implementation (e.g. TF-A)
  - Using existing SMC ranges (OEM, SiP, ToS) result in a lot of duplicity and fragmentation
- SMCCC 1.5 introduced EL3 vendor specific monitor range [4]
  - EL3 FW may standardize/document the services in the range
  - Better to be used by tightly integrated SW (e.g. projects at trustedfirmware.org )
- TF-A has created EL3 range for its usage
  - Initial migration of DebugFS and Performance Measurement Framework (PMF) from Arm SiP range
  - Other use cases
    - Secure OS to request memory map from BL31?
    - Intentional panic in TF-A for test automation

# Feature detection

- Architecture feature states ENABLE_FEAT_XXX
  - DISABLED (0): Feature **is** disabled statically at compile time.
  - ALWAYS (1): Feature **is** enabled unconditionally at compile time.
  - CHECK (2): Feature **is** enabled, but checked at runtime
- Feature state DISABLE/ALWAYS ideal for fixed CPU platforms
  - Use FEATURE_DETECTION debug feature during development
    - Panic if feature enabled in FW and not present in CPU
- Feature state CHECK is ideal for platforms which support multiple different CPUs in different configurations (e.g. FVP)
  - Higher memory footprint
  - ID register check at runtime

# Enabling Architecture features

- Earlier ARM_ARCH_(MAJOR/MINOR) had dual purpose
  - Enable the architecture features
  - "march" option to be used by compiler
  - Because of this dependency Arch features can only be included up to which the compiler supports
- Fix this by decoupling ARM_ARCH_(MAJOR/MINOR) and use MARCH_DIRECTIVE
- All mandatory features will be enabled based on MAJOR/MINOR version
  - No need for platforms to enable mandatory features, Optional features still need to enabled.
  - Encourage platforms to set these values
  - Platform still can override mandatory features (discouraged)

# References


- [1] : https://trustedfirmware-a.readthedocs.io/en/latest/design_documents/context_mgmt_rework.html#introduction
- [2] : https://trustedfirmware-a.readthedocs.io/en/latest/components/ras.html
- [3] : https://github.com/FirmwareHandoff/firmware_handoff/
- [4] : https://developer.arm.com/documentation/den0028


LinaroConnect
MADRID 2024 | MAY 12-17 2024
Thank you