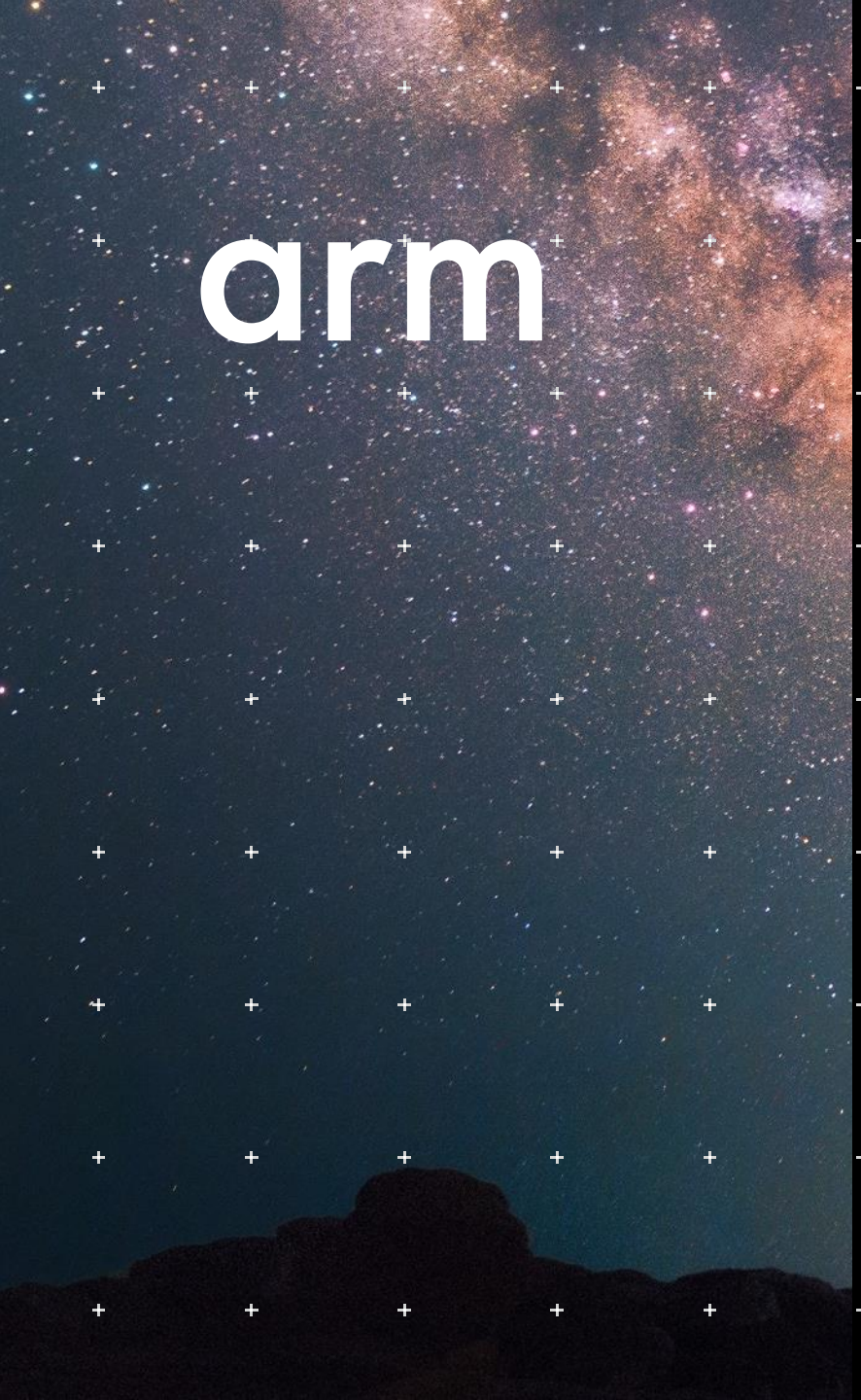


Errata ABI Design for EL3

Bipin/Sona

March 8, 2023

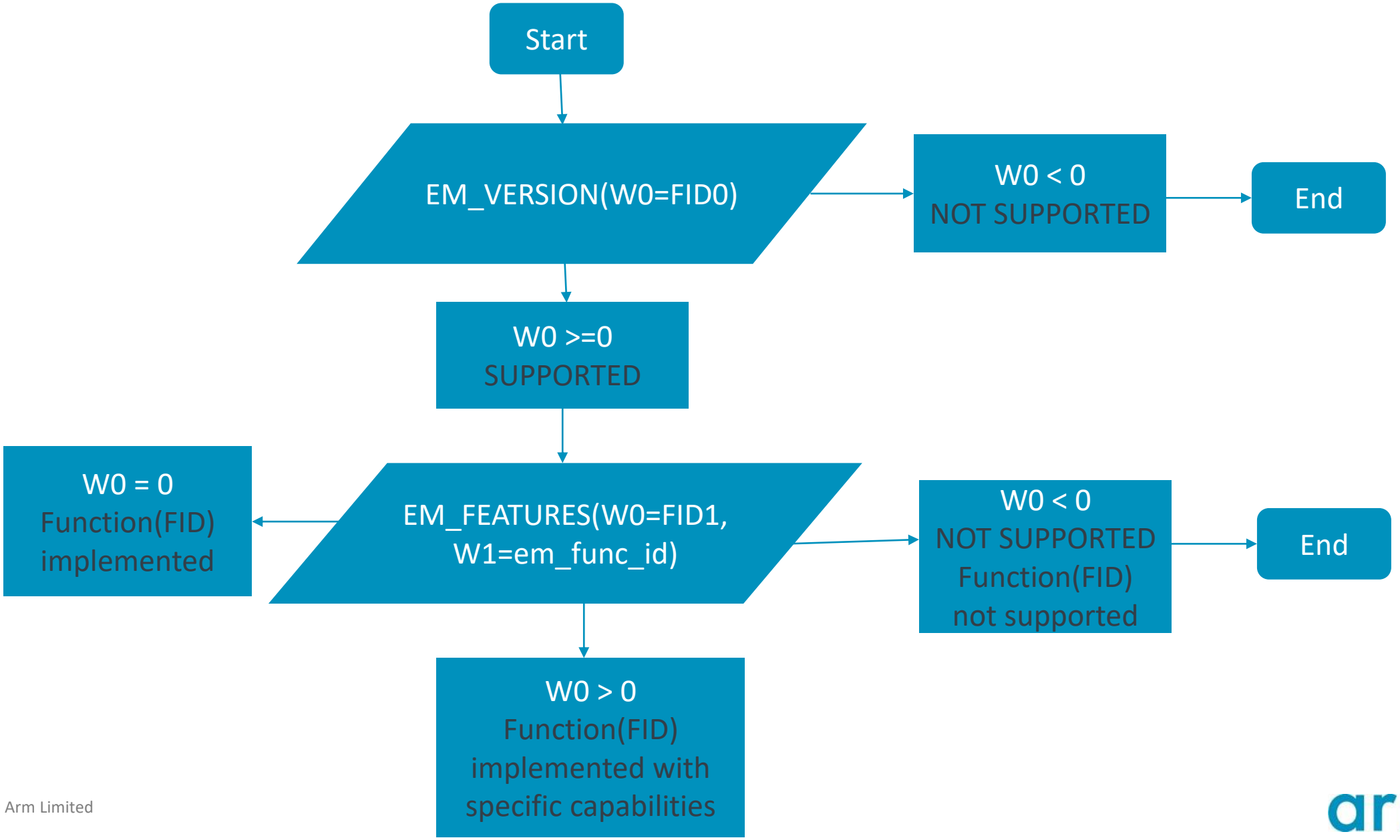
Agenda

- Brief introduction to Errata, Errata ABI
- Errata ABI calls supported by EL3
- Errata ABI discovery
- Pseudo code on how OS uses the errata ABI
- Design considerations
- ABI functions and inputs expectations
- Base Data Structure and CPU specific arrays
- High level sequence
- Implementation -> both generic and special case(non-arm interconnect IP)
- Code overview
- Testing

Errata ABI Calls supported by EL3

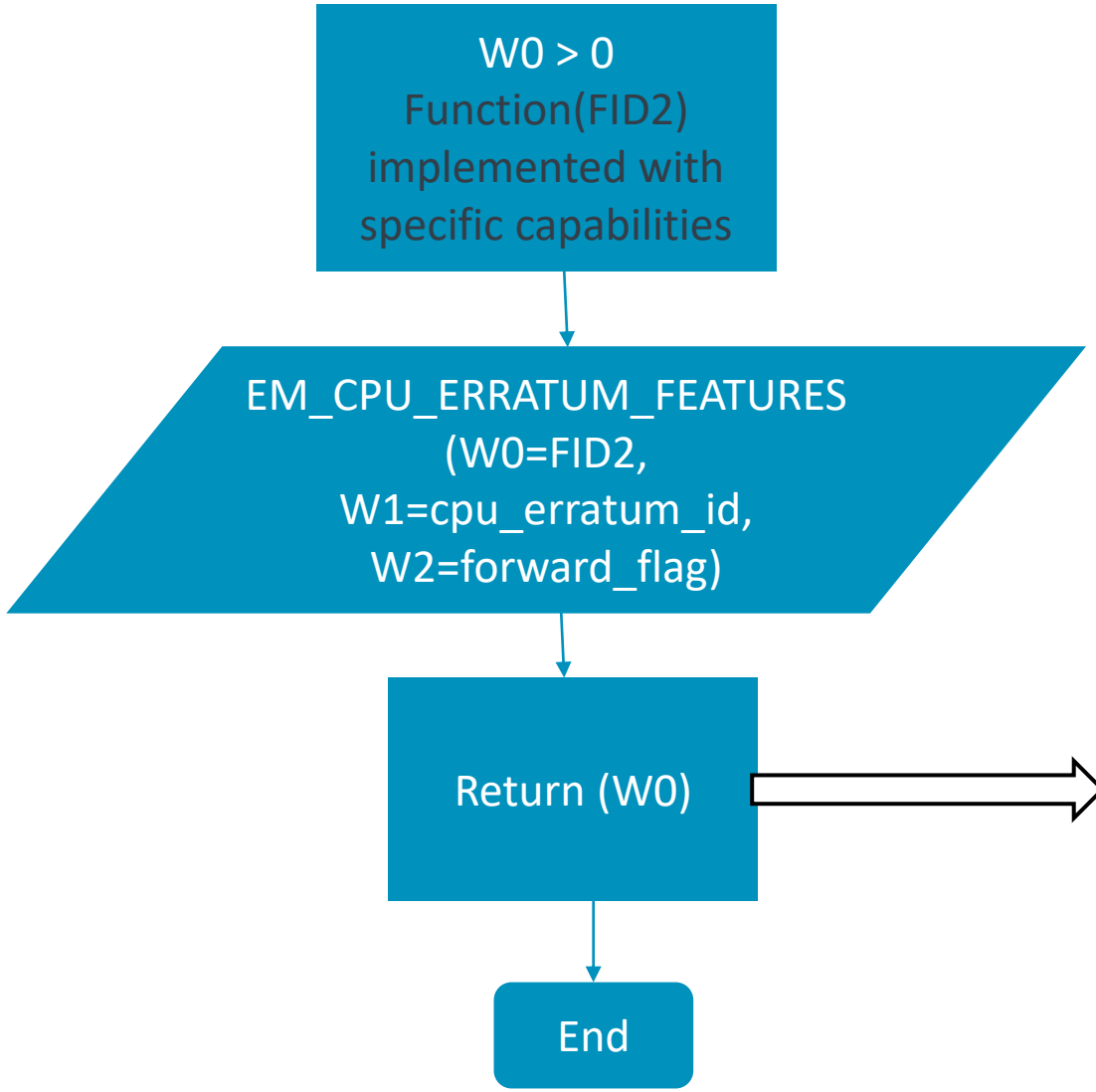
- CPU erratum is identified by the CPU_erratum_ID identifier, a unique 32-bit value that identifies the erratum on the specific CPU .
- The errata ABI complies with SMCCCv1.1 calling convention or higher
- Support for all the following ABI functions are mandatory from Errata ABI spec v1.0, with lower EL being the caller, will need to support from within EL3.
- The following functions are called using SMC
  - EM_VERSION (FID = 0x8400_00F0 = FID0)
  - EM_FEATURES (FID= 0x8400_00F1 = FID1)
  - EM_CPU_ERRATUM_FEATURES (FID =0x8400_00F2 = FID2)

ABI Discovery, Caller from lower EL (e.g. EL1)



arm

ABI Discovery(Cont..)



The return value of EM_CPU_ERRATUM_FEATURES is valid only for the calling CPU, the call must be performed on each CPU that the OS knows can be affected by a particular erratum.

Name	Value
HIGHER_EL_MITIGATION	3
NOT_AFFECTED	2
AFFECTED	1
SUCCESS	0
NOT_SUPPORTED	-1
INVALID_PARAMETERS	-2
UNKNOWN_ERRATUM	-3

Return codes for EM_CPU_ERRATUM_FEATURES()

arm

How OS uses the ABI?

```
bool need_cpu_erratum_local_wa(u32 cpu_erratum_id_list[], int num_erratum_entries)
{
        int forward_flag = 0;
        for (int idx = 0; idx < num_erratum_entries; idx++) {

                u32 cpu_erratum_id = cpu_erratum_id_list[idx];
                int ret = smccc_call(EM_CPU_ERRATUM_FEATURES, cpu_erratum_id, forward_flag);

                switch (ret) {
                  case EM_HIGHER_EL_MITIGATION:
                  case EM_NOT_AFFECTED:          // Return value when the erratum has been mitigated in hardware
                        return false;
                  case EM_UNKNOWN_ERRATUM: // Firmware does not recognise the cpu_erratum_id on this CPU.
                        continue;                // OS may decide to implement the workaround if applicable
                  case EM_AFFECTED:  // The CPU is affected by the erratum, the OS should deploy a workaround.
                        return true;
                }
        }
        return true;
}
```

Design considerations for Errata ABI support

- Inputs
  - Inputs to the ABI
    - All inputs for the ABI functions received through SMC conduit
    - MIDR value to be read (e.g. read_MIDR_EL1)
  - Build/Compile time options
    - CPUs in the platform
      - to compile applicable pre-initialized arrays
- Data structure needed & sizing
- No dynamic memory allocation
- Static initialization of data structures
- Performance/memory size

Design Considerations

- CPU erratum data structures will be populated statically for every CPU(which has one or more erratum) supported in TF-A
- This will introduce a new file which need to be populated with every errata implementation to add an extra entry in the CPU specific array of data structures
- All erratum IDs applicable for a given CPU will be populated in ascending order in the newly introduced array of structures. This enables binary search to look up for the erratum ID within the array
- Assume the data structure arrays are compiled in, based on applicable CPUs within the platform, using build flags
- Current ABI design doesn’t consider Split errata that is mitigated in multiple ELs, but the expectation is it could be supported with minimum changes to the data structure if it is supported in future
- For return of UNKNOWN_ERRATA(-3) for EM_CPU_ERRATUM_FEATURES, OS will implement erratum if applicable. This is because TF-A doesn’t hold all Cat B errata info for every CPU so far but only the ones with the mitigation in EL3. So, firmware won’t have the full list.

ABI Functions and Input Expectations for SMC call

- The following functions are called using SMC and hence the incoming register values are expected as mentioned in the ABI spec

  - EM_VERSION (W0=FID=0x8400_00F0)

  - EM_FEATURES (W0 = FID=0x8400_00F1,
    W1=em_func_id = 0x8400_00F2) // Currently the only supported erratum features function is EM_CPU_ERRATUM_FEATURES, any other em_func_id value will return “NOT_SUPPORTED”

  - EM_CPU_ERRATUM_FEATURES (W0=FID=0x8400_00F2,
    W1=CPU_erratum_ID, // Erratum ID
    W2=forward_flag) // Forward flag, MBZ when called from EL1

High level sequence

1. Build flow includes the data structure for CPUs in the platform

2. Part of the initialization in BL31, the data structure is initialized to external memory

3. After control is transferred to EL1, at some point OS does the discovery process , if the ABI is supported (as mentioned in earlier flow chart)

4. If the ABI is supported, OS calls EM_CPU_ERRATUM_FEATURES with the errata ID

5. Based on the MIDR of the calling CPU and errata ID, do a binary search in the correct CPU data structure array to determine if the errata ID is recognized

6. If errata ID matches, compare the version for which errata is applicable and do the appropriate return status

7. OS uses the return value and take appropriate action. e.g., applying the mitigation for errata that needs to be applied at a lower EL

8. Repeat steps 4 to 7 for all applicable erratas.

Data Structure

```
struct em_cpu {
   unsigned int em_errata_id;
   unsigned char em_rxpx_lo;         // Lowest revision errata is applicable e.g. r0p1 = 0x1, r0p1 = 0x1, r1p2 =0x12
   unsigned  char em_rxpx_hi;        // Highest revision errata is applicable e.g. r0p1 = 0x1, r0p1 = 0x1, r1p2 =0x12
   unsigned char hardware_mitigated; // version number if erratum is fixed in hardware
   bool hw_flag;                          // flag to indicate erratum is fixed in hardware
   bool arm_interconnect;            // Flag to indicate if platform uses arm or non-arm interconnect
   bool platform_affected;           // Flag to indicate if platform is affected or not
};
```

e.g., Use the above data structure to build an array of struct for each CPU to include all applicable errata.

```
struct em_cpu_list{
            unsigned long cpu_pn;              /* field to hold cpu specific part number defined in midr reg*/
            struct   em_cpu cpu_errata_list[MAX_SIZE];
};
```

- struct em_cpu_list cpu_list[] = {

```
#if CORTEX_A78_H_INC
{
  .cpu_pn = CORTEX_A78_MIDR,
  .cpu_errata_list = {
                {1688305, 0x00, 0x10},
                {1821534, 0x00, 0x10},
                {2395406, 0x00, 0x12},
#if ERRATA_NON_ARM_INTERCONNECT
                {2712571, 0x00, 0x12, 0x00, false, \
                ERRATA_NON_ARM_INTERCONNECT, ERRATA_A78_2712571},
},
#endif
.
.
…. ……..N such cpu entries
};
```

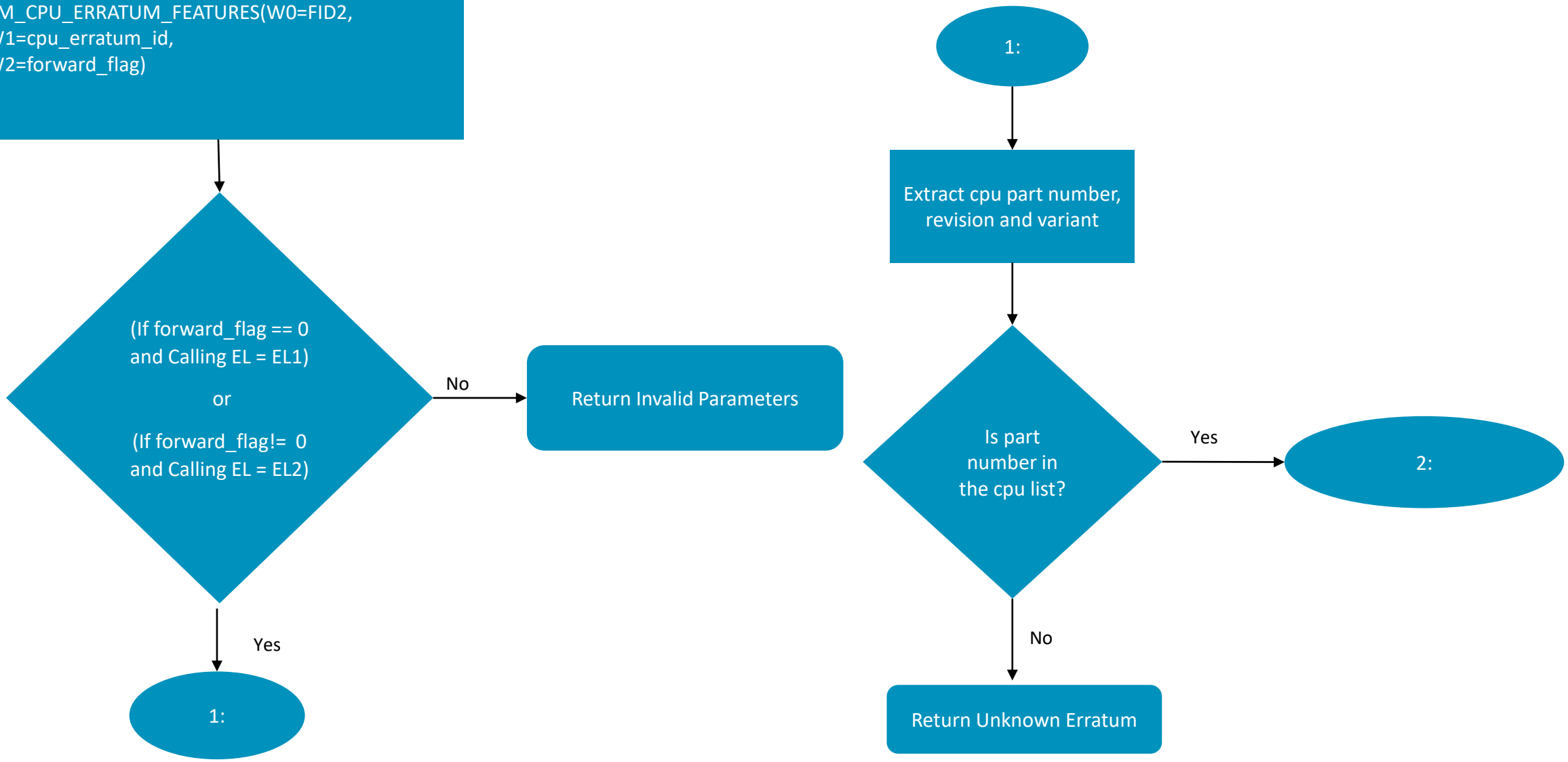
Flag enabled/disabled in platform make file

Flag to indicate non arm interconnect IP.

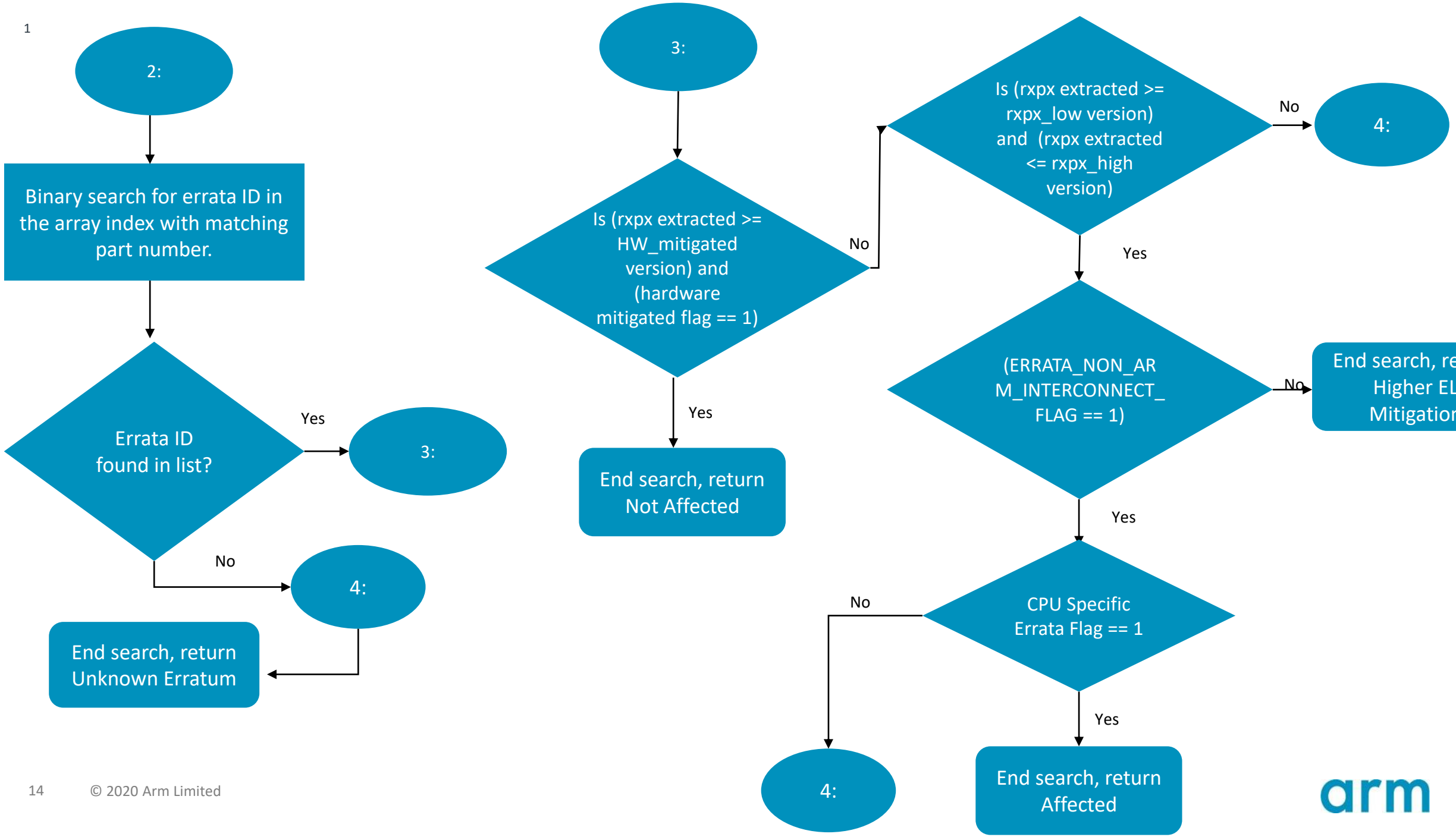
Refer : https://review.trustedfirmware.org/c/TF-A/trusted-firmware-a/+/19835/1/services/std_svc/errata_abi/errata_abi_main.c

High level sequence flow chart

- EM_CPU_ERRATUM_FEATURES(W0=FID2,
  W1=cpu_erratum_id,
  W2=forward_flag)



1



Implementation

- Generic implementation:
- Link to the implementation : https://review.trustedfirmware.org/c/TF-A/trusted-firmware-a/+/19835

  -> Build flag to include feature -> ERRATA_ABI_SUPPORT = 1

- Special case implementation:
  - -> Build flag for platform's that do not have an arm interconnect -> ERRATA_NON_ARM_INTERCONNECT = 1
  - -> Specific flags for non-arm interconnect IP's, these flags can be enabled in the platform make file, based on whether the specific cpu errata needs to be enabled or not. These errata are not implemented in EL3.
  - -> Currently around 9+ cpu erratas included.

Array of CPU structures

- Refer : https://review.trustedfirmware.org/c/TF-A/trusted-firmware-a/+/19835/1/services/std_svc/errata_abi/errata_abi_main.c
- …/platform.mk · Gerrit Code Review (trustedfirmware.org)

```
struct em_cpu_list cpu_list[] = {
#if CORTEX_A9_H_INC
{
    .cpu_pn = CORTEX_A9_MIDR,
    .cpu_errata_list = {
        {794073, 0x00, 0xFF},
        {UINT_MAX},{UINT_MAX},
        {UINT_MAX},{UINT_MAX},
        {UINT_MAX},{UINT_MAX},
        {UINT_MAX},{UINT_MAX},
        {UINT_MAX},{UINT_MAX},
        {UINT_MAX},{UINT_MAX}
    }
},
#endif

#if CORTEX_A15_H_INC
{
    .cpu_pn = CORTEX_A15_MIDR,
    .cpu_errata_list = {
        {816470, 0x30, 0xFF},
        {827671, 0x30, 0xFF},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}
    }
},
#endif
```

```
ifeq (${HW_ASSISTED_COHERENCY}, 0)
# Cores used without DSU
        FVP_CPU_LIBS    +=      lib/cpus/aarch64/cortex_a35.S
                                lib/cpus/aarch64/cortex_a53.S
                                lib/cpus/aarch64/cortex_a57.S
                                lib/cpus/aarch64/cortex_a72.S
                                lib/cpus/aarch64/cortex_a73.S
else
# Cores used with DSU only
        ifeq (${CTX_INCLUDE_AARCH32_REGS}, 0)
        # AArch64-only cores
                FVP_CPU_LIBS    +=      lib/cpus/aarch64/cortex_a76.S
                                        lib/cpus/aarch64/cortex_a76ae.S
                                        lib/cpus/aarch64/cortex_a77.S
                                        lib/cpus/aarch64/cortex_a78.S
                                        lib/cpus/aarch64/neoverse_n_common.S
                                        lib/cpus/aarch64/neoverse_n1.S
                                        lib/cpus/aarch64/neoverse_n2.S
                                        lib/cpus/aarch64/neoverse_e1.S
                                        lib/cpus/aarch64/neoverse_v1.S
                                        lib/cpus/aarch64/neoverse_v2.S  \
                                        lib/cpus/aarch64/cortex_a78_ae.S
                                        lib/cpus/aarch64/cortex_a510.S
                                        lib/cpus/aarch64/cortex_a710.S
                                        lib/cpus/aarch64/cortex_a715.S
                                        lib/cpus/aarch64/cortex_x3.S
                                        lib/cpus/aarch64/cortex_a65.S
                                        lib/cpus/aarch64/cortex_a65ae.S
                                        lib/cpus/aarch64/cortex_a78c.S
                                        lib/cpus/aarch64/cortex_hayes.S
                                        lib/cpus/aarch64/cortex_hunter.S
                                        lib/cpus/aarch64/cortex_hunter_elp_arm
                                        lib/cpus/aarch64/cortex_x2.S
                                        lib/cpus/aarch64/neoverse_poseidon.S
```

```
ifeq (${ERRATA_ABI_SUPPORT}, 1)
# enable the cpu macros for errata abi interface
ifeq (${ARCH}, aarch64)
ifeq (${HW_ASSISTED_COHERENCY}, 0)
CORTEX_A35_H_INC        := 1
CORTEX_A53_H_INC        := 1
CORTEX_A57_H_INC        := 1
CORTEX_A72_H_INC        := 1
CORTEX_A73_H_INC        := 1
$(eval $(call add_define, CORTEX_A35_H_INC))
$(eval $(call add_define, CORTEX_A53_H_INC))
$(eval $(call add_define, CORTEX_A57_H_INC))
$(eval $(call add_define, CORTEX_A72_H_INC))
$(eval $(call add_define, CORTEX_A73_H_INC))
else
ifeq (${CTX_INCLUDE_AARCH32_REGS}, 0)
CORTEX_A76_H_INC        := 1
CORTEX_A77_H_INC        := 1
NEOVERSE_N1_H_INC       := 1
CORTEX_A78_AE_H_INC     := 1
CORTEX_A510_H_INC       := 1
CORTEX_A710_H_INC       := 1
CORTEX_A78C_H_INC       := 1
CORTEX_X2_H_INC         := 1
CORTEX_A55_H_INC        := 1
CORTEX_A75_H_INC        := 1
$(eval $(call add_define, CORTEX_A76_H_INC))
$(eval $(call add_define, CORTEX_A77_H_INC))
$(eval $(call add_define, CORTEX_A78_AE_H_INC))
```

Platform.mk file -> FVP

Non-Arm Interconnect flags

```
# Add errata for Demeter when a non-arm interconnect is enabled.
ifeq ($(ERRATA_NON_ARM_INTERCONNECT), 1)
ERRATA_A710_2701952     :=1     #placeholder for cortex A710
ERRATA_A78_2712571      :=1     #placeholder for cortex A78
ERRATA_A78C_2712575     :=1     #placeholder for cortex A78C
ERRATA_A78_AE_2712574   :=1     #placeholder for cortex A78_AE
ERRATA_V2_2719103       :=1     #placeholder for neoverse V2(demeter)
ERRATA_A715_2701951     :=1     #placeholder for cortex A715(makalu)
ERRATA_X2_2701952       :=1     #placeholder for cortex X2(matterhorn)
ERRATA_N2_2728475       :=1     #placeholder for neoverse N2(perseus)
ERRATA_V1_2701953       :=1     #placeholder for neoverse V1(zeus)
else
ERRATA_A710_2701952     :=0
ERRATA_A78_2712571      :=0
ERRATA_A78C_2712575     :=0
ERRATA_A78_AE_2712574   :=0
ERRATA_V2_2719103       :=0
ERRATA_A715_2701951     :=0
ERRATA_X2_2701952       :=0
ERRATA_N2_2728475       :=0
ERRATA_V1_2701953       :=0
endif
```

Platform.mk file -> FVP

Cpu_list

```
#if CORTEX_A715_H_INC
{
        .cpu_pn = CORTEX_MAKALU_MIDR,
        .cpu_errata_list = {
        #if ERRATA_NON_ARM_INTERCONNECT
        {2701951, 0x00, 0x11, 0x00, false, \
        ERRATA_NON_ARM_INTERCONNECT, ERRATA_A715_2701951},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX}
        #endif
        }
},
#endif
```


arm

List of errata's that affect platforms with non-arm interconnect

Makalu-ELP / Cortex-A715 - 2701951
- Arm Cortex-X3 (MP141) Software Developer Errata Notice

Demeter / Neoverse V2 - 2719103
- Arm Neoverse V2 (MP158) Software Developer Errata Notice

Matterhorn / Cortex-A710 - 2701952
- Arm Cortex-A710 (MP117) Software Developer Errata Notice

Matterhorn-ELP / Cortex-X2 - 2701952
- Arm Cortex-X2 (MP121) Software Developer Errata Notice

Perseus/Neoverse N2 - 2728475
- Arm Neoverse N2 (MP128) Software Developer Errata Notice

Zeus / Neoverse V1 - 2701953
- Arm Neoverse V1 (MP076) Software Developer Errata Notice

Hercules / Cortex-A78 - 2712571
- Arm Cortex-A78 (MP102) Software Developer Errata Notice

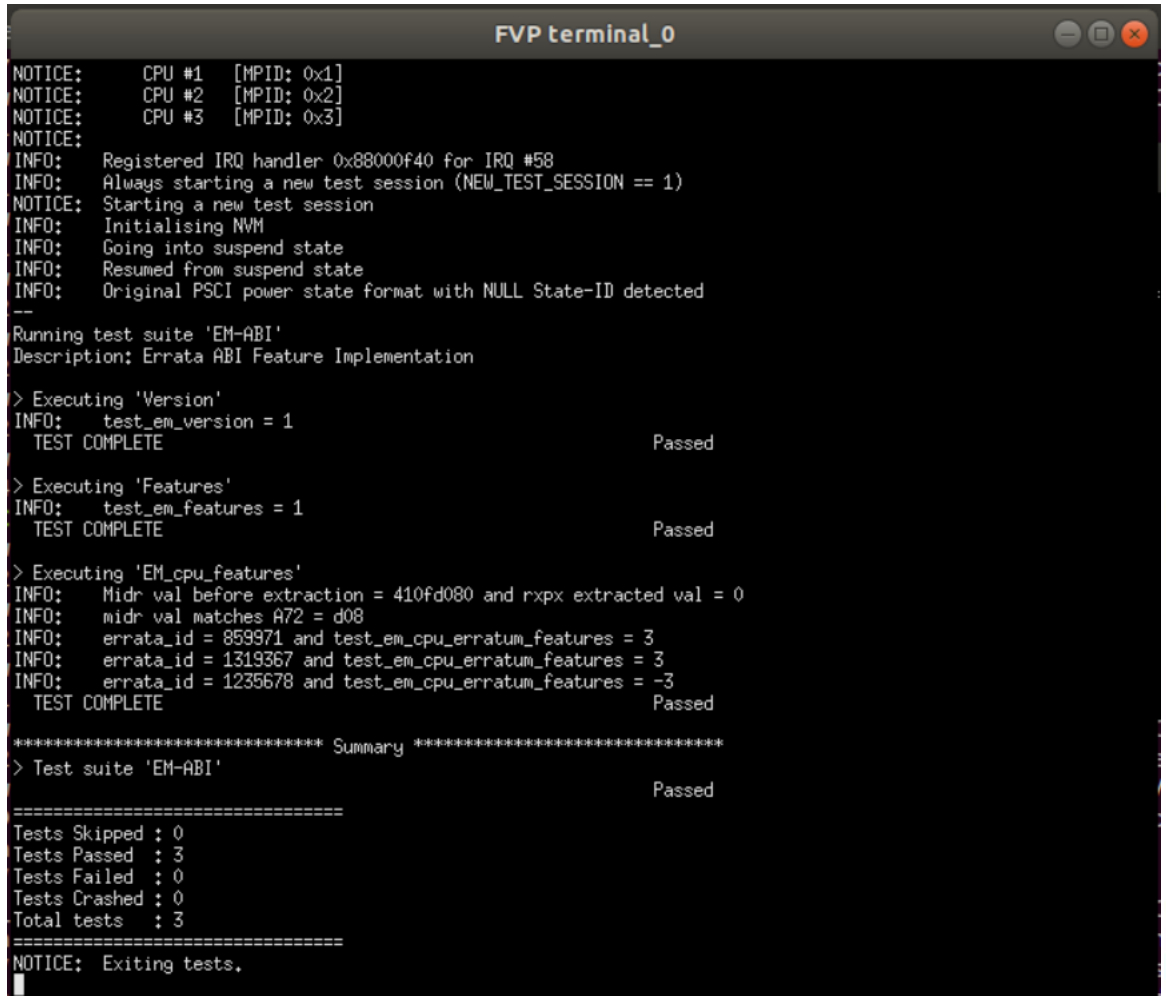
Hercules-AE / Cortex-A78AE - 2712574
- Arm Cortex-A78AE (MP105) Software Developer Errata Notice

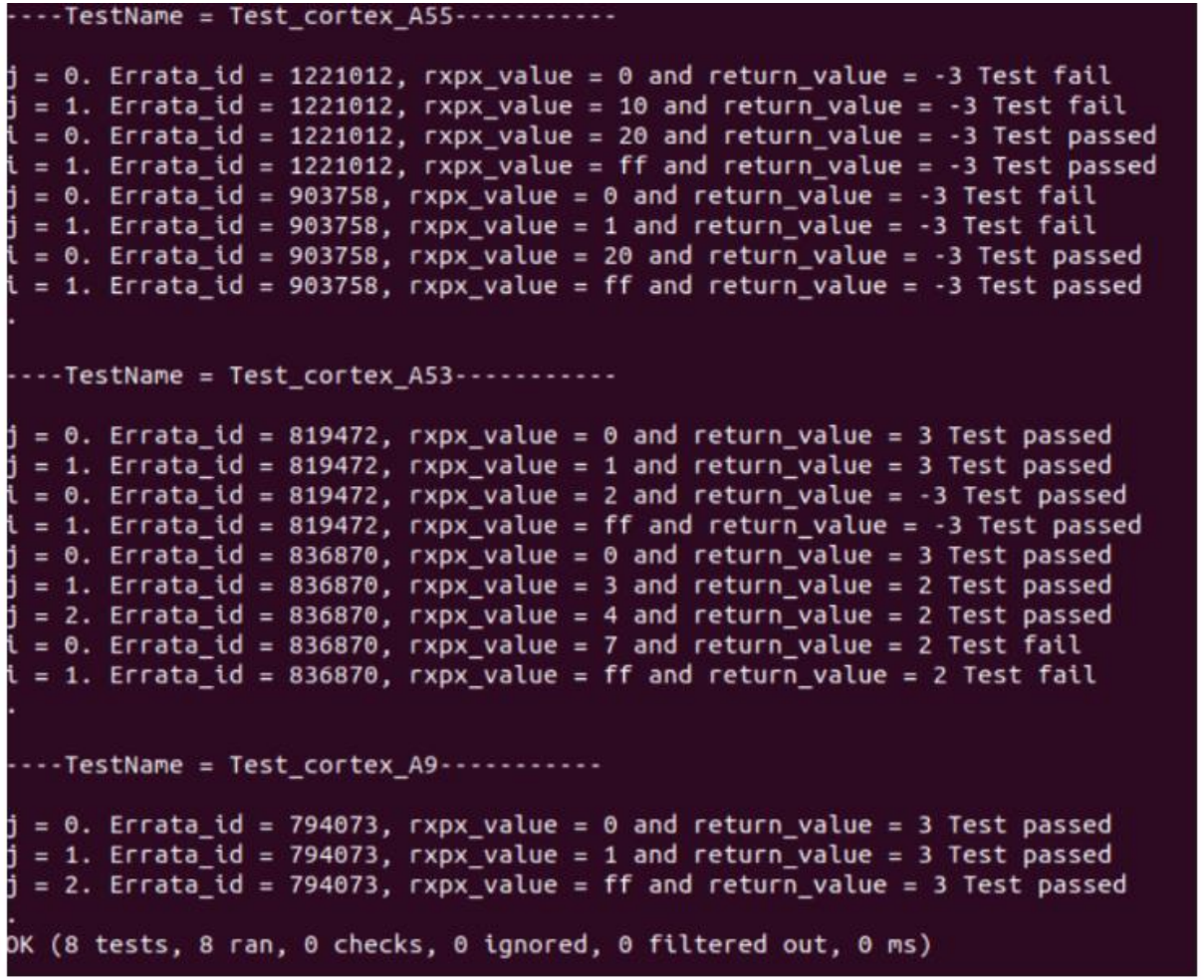
HerculesPrime / Cortex-A78C - 2712575
- Arm Cortex-A78C (MP154) Software Developer Errata Notice

Hera / HeraPrime
- Not Supported in TF-A

Testing

- Unit Testing : https://gerrit.oss.arm.com/c/trusted-firmware/tf-a-unit-tests/+/252281
- TFTF Testing : https://gerrit.oss.arm.com/c/trusted-firmware/tf-a-tests/+/245954





arm

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