Errata implementation framework

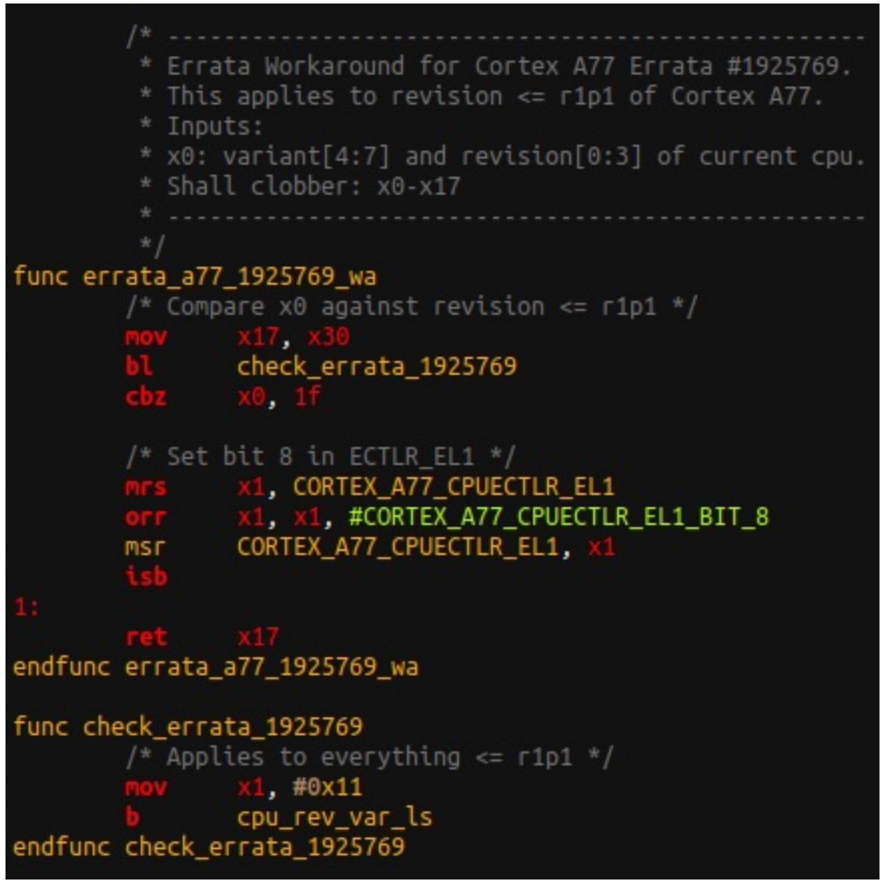
Boyan Karatotev (boyan.karatotev@arm.com)
09.03.2023

Motivation

1. Implementing an errata today is verbose

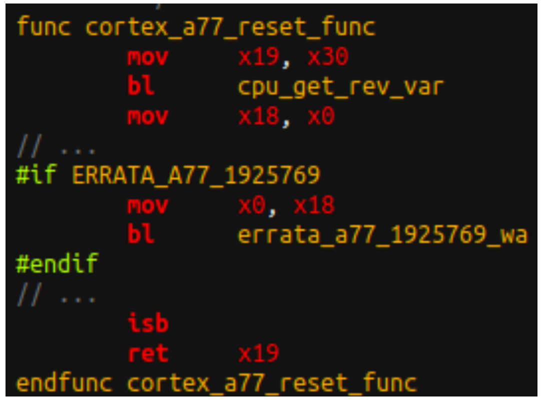
Majority set a bit at reset

a)



```
        /* ------------------------------------------------
         * Errata Workaround for Cortex A77 Errata #1925769.
         * This applies to revision <= r1p1 of Cortex A77.
         * Inputs:
         * x0: variant[4:7] and revision[0:3] of current cpu.
         * Shall clobber: x0-x17
         * ------------------------------------------------
         */
func errata_a77_1925769_wa
        /* Compare x0 against revision <= r1p1 */
        mov     x17, x30
        bl      check_errata_1925769
        cbz     x0, 1f

        /* Set bit 8 in ECTLR_EL1 */
        mrs     x1, CORTEX_A77_CPUECTLR_EL1
        orr     x1, x1, #CORTEX_A77_CPUECTLR_EL1_BIT_8
        msr     CORTEX_A77_CPUECTLR_EL1, x1
        isb
1:
        ret     x17
endfunc errata_a77_1925769_wa

func check_errata_1925769
        /* Applies to everything <= r1p1 */
        mov     x1, #0x11
        b       cpu_rev_var_ls
endfunc check_errata_1925769
```

b)

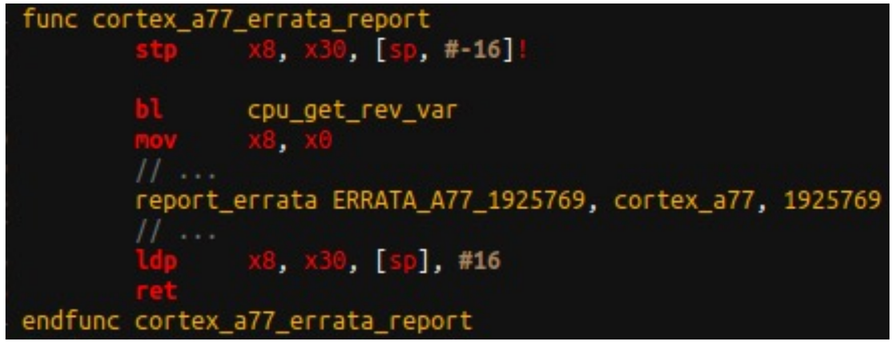


```
func cortex_a77_reset_func
        mov     x19, x30
        bl      cpu_get_rev_var
        mov     x18, x0
// ...
#if ERRATA_A77_1925769
        mov     x0, x18
        bl      errata_a77_1925769_wa
#endif
// ...
        isb
        ret     x19
endfunc cortex_a77_reset_func
```

+ make rule

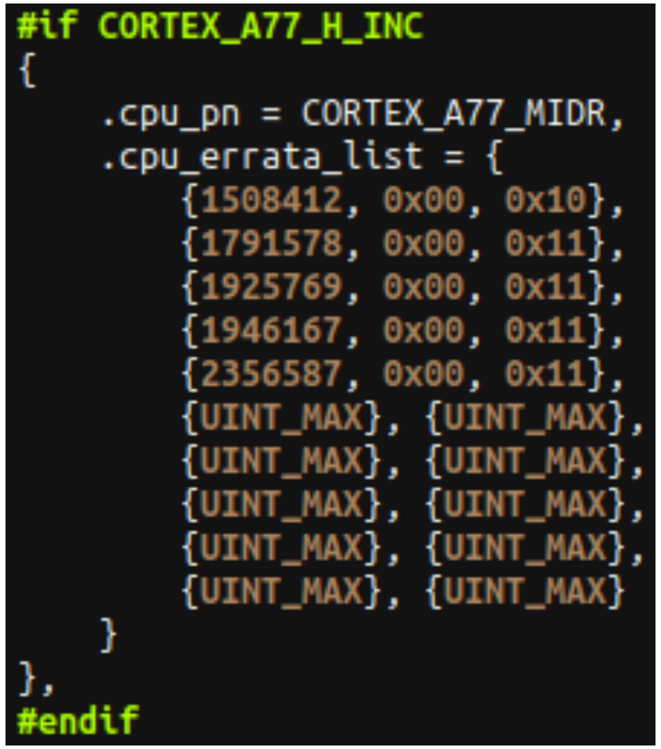
+ docs mention

c)



```
func cortex_a77_errata_report
        stp     x8, x30, [sp, #-16]!

        bl      cpu_get_rev_var
        mov     x8, x0
        // ...
        report_errata ERRATA_A77_1925769, cortex_a77, 1925769
        // ...
        ldp     x8, x30, [sp], #16
        ret
endfunc cortex_a77_errata_report
```

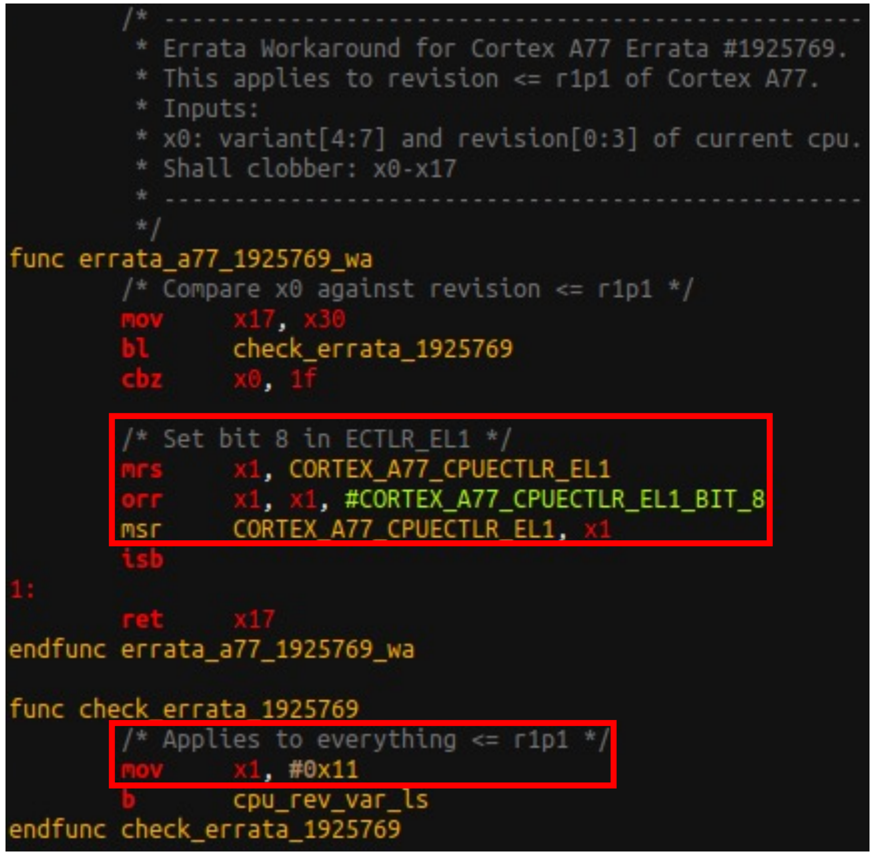
2. The errata ABI



```
#if CORTEX_A77_H_INC
{
    .cpu_pn = CORTEX_A77_MIDR,
    .cpu_errata_list = {
        {1508412, 0x00, 0x10},
        {1791578, 0x00, 0x11},
        {1925769, 0x00, 0x11},
        {1946167, 0x00, 0x11},
        {2356587, 0x00, 0x11},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX},
        {UINT_MAX}, {UINT_MAX}
    }
},
#endif
```

- 1 more place to edit
- Information again redundant
- But not accessible

All the useful code



```
        /* ---------------------------------------------------
         * Errata Workaround for Cortex A77 Errata #1925769.
         * This applies to revision <= r1p1 of Cortex A77.
         * Inputs:
         * x0: variant[4:7] and revision[0:3] of current cpu.
         * Shall clobber: x0-x17
         * ---------------------------------------------------
         */
func errata_a77_1925769_wa
        /* Compare x0 against revision <= r1p1 */
        mov     x17, x30
        bl      check_errata_1925769
        cbz     x0, 1f

        /* Set bit 8 in ECTLR_EL1 */
        mrs     x1, CORTEX_A77_CPUECTLR_EL1
        orr     x1, x1, #CORTEX_A77_CPUECTLR_EL1_BIT_8
        msr     CORTEX_A77_CPUECTLR_EL1, x1
        isb
1:
        ret     x17
endfunc errata_a77_1925769_wa

func check_errata_1925769
        /* Applies to everything <= r1p1 */
        mov     x1, #0x11
        b       cpu_rev_var_ls
endfunc check_errata_1925769
```

The rest is boilerplate

And very annoying to get past review

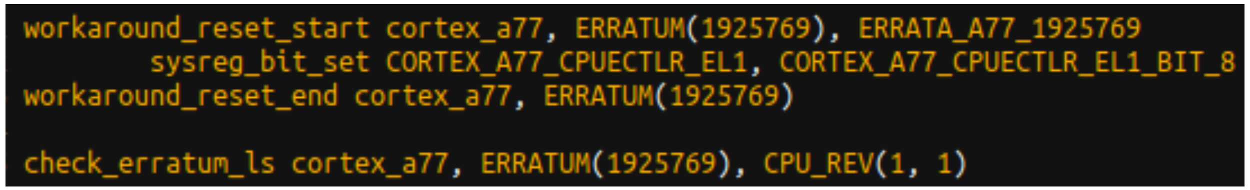
Of course, some are more involved

- Longer workaround sequence
- More involved rev check
- not applied at reset

Practically all errata can be pigeonholed to this template

With small provisions to account for variations

Proposal – Aarch64 erratum implementation



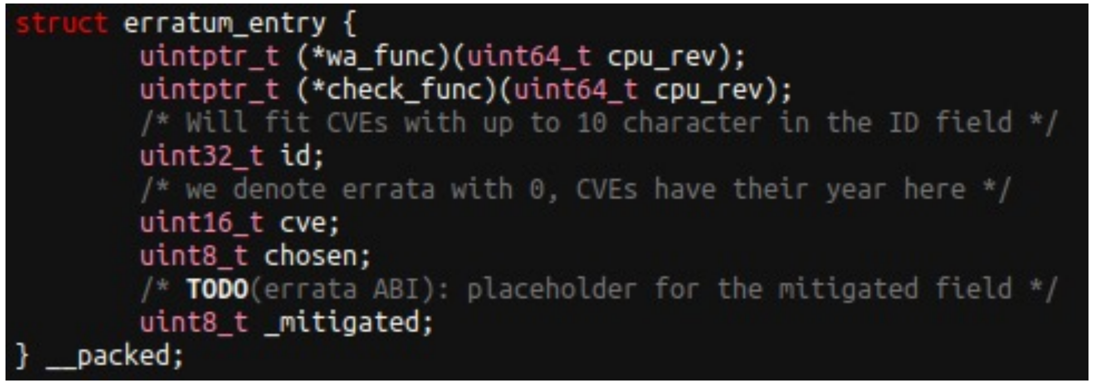
```
workaround_reset_start cortex_a77, ERRATUM(1925769), ERRATA_A77_1925769
        sysreg_bit_set CORTEX_A77_CPUECTLR_EL1, CORTEX_A77_CPUECTLR_EL1_BIT_8
workaround_reset_end cortex_a77, ERRATUM(1925769)

check_erratum_ls cortex_a77, ERRATUM(1925769), CPU_REV(1, 1)
```

+ make rule

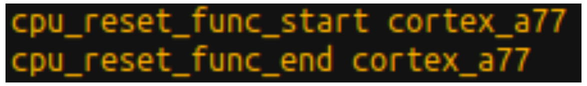
+ docs mention

workaround_reset_{start, end}	- wrapper of erratum workaround function
workaround_runtime_{start, end}	- same but workaround manually applied
sysreg_bit_set	- reads back and asserts bit set when DEBUG=1
check_erratum_{ls, hs, range}	- checker helper
A runtime	

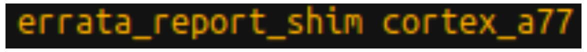
The runtime



```c
struct erratum_entry {
	uintptr_t (*wa_func)(uint64_t cpu_rev);
	uintptr_t (*check_func)(uint64_t cpu_rev);
	/* Will fit CVEs with up to 10 character in the ID field */
	uint32_t id;
	/* we denote errata with 0, CVEs have their year here */
	uint16_t cve;
	uint8_t chosen;
	/* TODO(errata ABI): placeholder for the mitigated field */
	uint8_t _mitigated;
} __packed;
```



```
cpu_reset_func_start cortex_a77
cpu_reset_func_end cortex_a77
```



```
errata_report_shim cortex_a77
```

- **workaround_*_start** registers an erratum entry
  - In per-cpu errata_entries section (like cpu_ops)
- **cpu_reset_func** applies selected ones from the list
  - Special cpu behaviour can happen after
- **errata_report_shim** does reporting when DEBUG=1
  - Common C function for all CPUs iterates the list

This covers the majority of cases

Each macro can be incrementally unraveled to the old method for particularly nasty errata

The Procedure Call Standard

- Some of the cpu operations must obey the PCS

- => obey the PCS throughout

- Based on the following (simplified) interpretation

Reg	Use
r0 - r15	Scratch registers. Anyone can use at any time
r16, r17	Avoid using. Used by the linker. Any branch (with a relocation) may corrupt it
r18	Avoid using. Scratch, but may be used by the platform for inter procedure call state. Is this us?
r19 - r28	Caller saved
r29, r30	FP, LR

Mandated register assignments

- to avoid having to do register management
  - Also will simplify implementation
- Subset of the full PCS
  - To eliminate the problem

function	register	treatment
Any BL	r0-r4	May clobber
Workaround implementation	r0-r7	May clobber
	r0, r5	Parameter to implementation - cpu_rev_var
Erratum checker function	r0-r4	May clobber
All other	r8 - r30	Treat as callee saved

- Runtime has similar assignments, documented in code

Aarch32

```
add_erratum_entry cortex_a57, ERRATUM(813420), ERRATA_A57_813420
```

- Implementation stays the same
- Only registered to the framework for debug and ABI reporting
- Removes some redundancy but little benefit to do fully

The implementation

Runtime part

Cost – workaround/check functions

- Check function identical
- Workaround function – practically identical
  - isb moved to reset_func
  - extra mov for compatibility
  - ASSERT when DEBUG=1 (gone on release builds)

Cost – errata_entries list

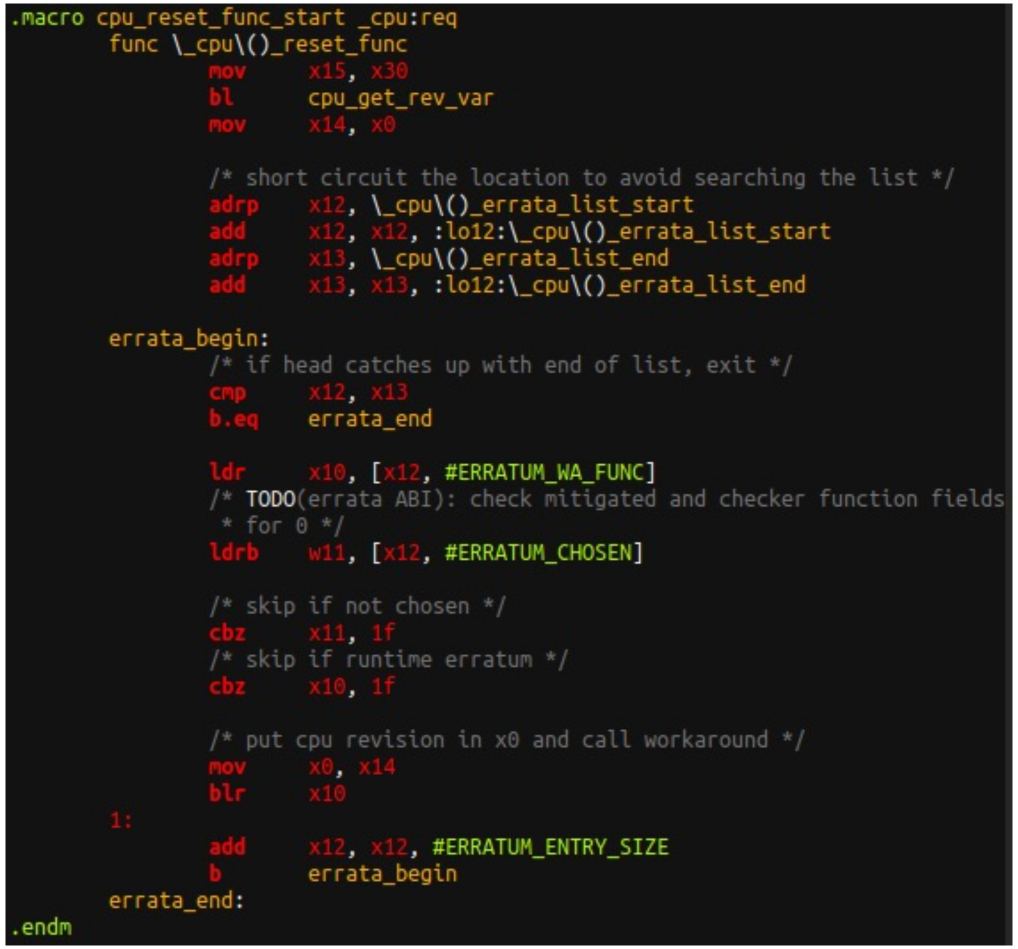
- Per-cpu list
- 24 bytes per entry, 1 entry per erratum
  - some overlap with errata ABI. Designed to be reused
  - Minimal information to enable runtime and ABI reporting

Cost – reset_func

- Fixed size of 19 instructions (76 bytes)
  - Previously 5 fixed + 2 per erratum

- Loop with 8 instructions per erratum
  - Runs even if disabled (previously compiled out)
  - Previously only 2

- Space saving when > 7 errata per cpu



```
.macro cpu_reset_func_start _cpu:req
        func \_cpu\()_reset_func
                mov     x15, x30
                bl      cpu_get_rev_var
                mov     x14, x0

                /* short circuit the location to avoid searching the list */
                adrp    x12, \_cpu\()_errata_list_start
                add     x12, x12, :lo12:\_cpu\()_errata_list_start
                adrp    x13, \_cpu\()_errata_list_end
                add     x13, x13, :lo12:\_cpu\()_errata_list_end

        errata_begin:
                /* if head catches up with end of list, exit */
                cmp     x12, x13
                b.eq    errata_end

                ldr     x10, [x12, #ERRATUM_WA_FUNC]
                /* TODO(errata ABI): check mitigated and checker function fields
                 * for 0 */
                ldrb    w11, [x12, #ERRATUM_CHOSEN]

                /* skip if not chosen */
                cbz     x11, 1f
                /* skip if runtime erratum */
                cbz     x10, 1f

                /* put cpu revision in x0 and call workaround */
                mov     x0, x14
                blr     x10
        1:
                add     x12, x12, #ERRATUM_ENTRY_SIZE
                b       errata_begin
        errata_end:
.endm
```

- disabled errata are not left out of the list due to the errata ABI

Cost – errata reporting

- **A debug feature**. Compiled out on release builds
  - Optimality superseded by ease of use
- Common print function in C
  - Around 250 instructions
- Per-cpu shim
  - 5 instructions

Going forward

The migration

- Old and new style interoperable
  - old is inaccessible to framework

- Migrate every aarch64 cpu - 3 patches

- Fill-in every aarch32 cpu - 1 patch

- Converge with errata ABI

- Gradually submit to LTS

Aarch64 correctness

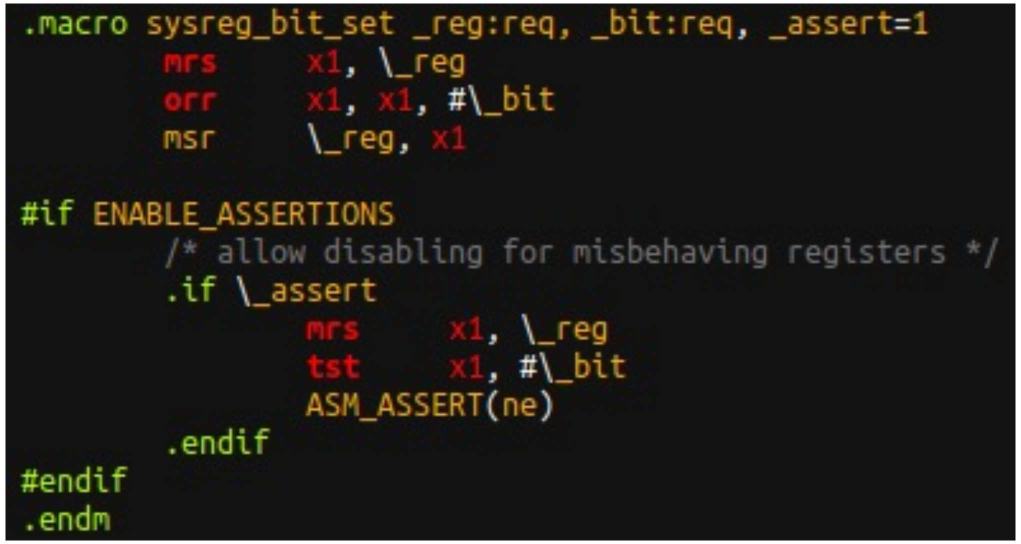
- Patch 1 – reorder only
  - To enforce reporting and binary search requirements
- Patch 2 – remove boilerplate and register to framework
  - Retain git blame of actual workaround
- Patch 3 – move to bit setting helpers
  - Readability and consistency benefit
  - Strictly speaking optional

- Script to verify identical binary result
  - Within established tolerances eg. missing isb
- Manual debugger run
  - a few will need genuine refactors
  - i.e. the usual errata testing process

Correctness contd.

- Open question - build workarounds for CI runs
  - no elegant solution was apparent
  - Only done for Juno (hard-coded)

- Add asserts – location open question
  - Currently sysreg_bit_set reads the bit back
  - Assert workaround ran? How?
  - Assert get_cpu_var ran? How?



```
.macro sysreg_bit_set _reg:req, _bit:req, _assert=1
        mrs     x1, \_reg
        orr     x1, x1, #\_bit
        msr     \_reg, x1

#if ENABLE_ASSERTIONS
        /* allow disabling for misbehaving registers */
        .if \_assert
                mrs     x1, \_reg
                tst     x1, #\_bit
                ASM_ASSERT(ne)
        .endif
#endif
.endm
```

- Aarch32 avoids correctness since no refactoring there

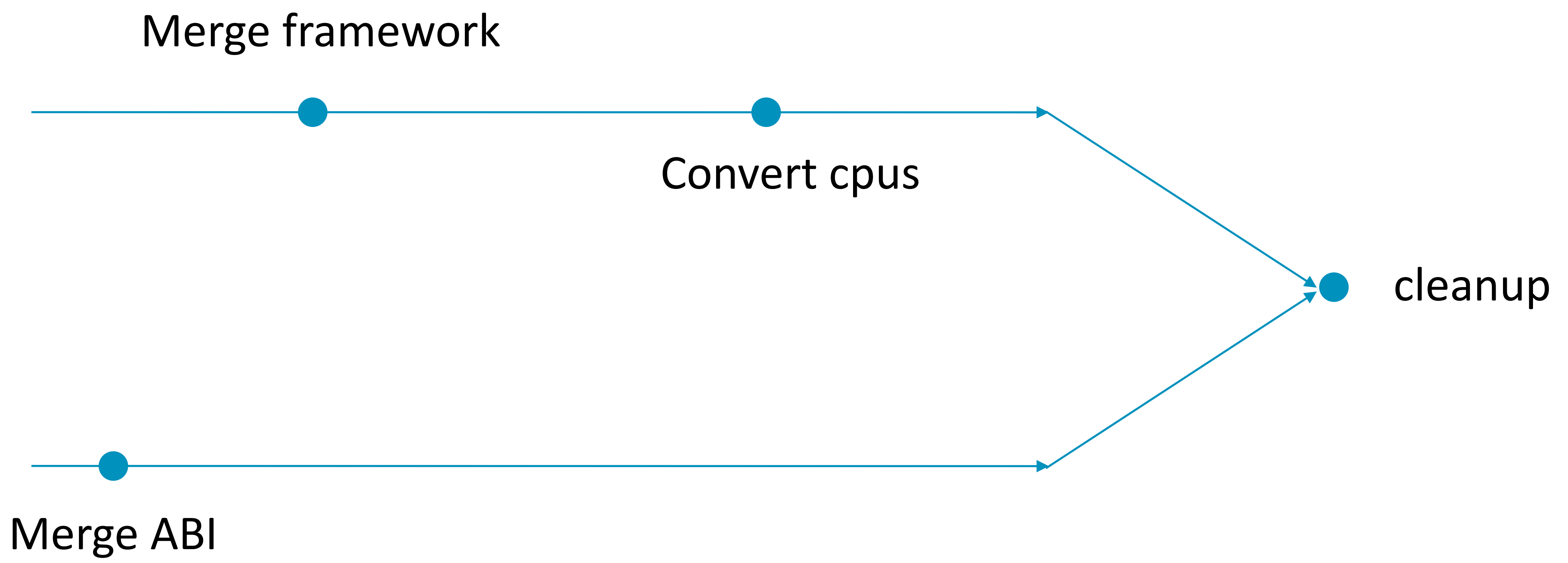
Downstream errata

- **Please submit upstream**
  - We will do migration work for each CPU
- Changes easy to implement
  - But will assist
- Platform errata unaffected, but inaccessible to framework and ABI

LTS

- Patches can be submitted to LTS
- LTS identical to master, no changes required
  - For errata, at least

Errata ABI convergence



Code

https://review.trustedfirmware.org/q/topic:%22bk%252Ferrata_refactor%22+

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