Document Number: IMX53CE
Rev. 4, 06/2013

Chip Errata for the i.MX53

This document details the silicon errata known at the time of publication for the i.MX53 multimedia applications processors. The contents of this errata apply to the following devices: IMX53xA, IMX53xC, and IMX53xD.

Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Changes
4	06/2013	• Added LDO erratum ERR007080 • Updated SRTC erratum ENGcm12374 • Updated software workaround for ESDCTRLv2 erratum ENGcm12377 • Updated silicon fix for SCC erratum ENGcm12354
3	03/2013	• Added: – SRTC erratum ENGcm12374 – ESAI erratum ENGcm12385 – ETB erratum ENGcm12386 – USB erratum ERR006308 • Removed CSPI ENGcm11554
2	01/2012	• Added eSDHC erratum ENGcm12290 • Added eSDHC erratum ENGcm12360 • Added SCC erratum ENGcm12354 • Added NFC erratum ENGcm12363 • Added eSDCTL erratum ENGcm12377 • Added EIM erratum ENGcm12379

freescale™
semiconductor

Table 1. Document Revision History (continued)

Rev. Number	Date	Substantive Changes
1	05/2011	• Updated description of ENGcm11851 • Updated solution for ENGcm11856
0	02/2011	• Initial public release.

Table 2 provides a cross-reference to match the revision code to the revision level marked on the device.

Table 2. Revision Level to Part Marking Cross-Reference

Revision	Package	Device Marking	Mask Set
MCIMX53 2.1	19 x 19 mm	MCIMX534AVV8C	N78C
MCIMX53 2.1	19 x 19 mm	MCIMX535DVV1C	N78C
MCIMX53 2.1	19 x 19 mm	MCIMX536AVV8C	N78C
MCIMX53 2.1	19 x 19 mm	MCIMX537CVV8C	N78C
MCIMX53 2.1	12 x 12 mm	SCIMX538DZK1C	N78C

The following table summarizes errata on the i.MX53 catergorized by module.

Table 3. Summary of Silicon Errata

Errata	Name	Solution	Page
AIPS			
ENGcm11136	AIPS: Unaligned access causes abort on writes to the internal registers	No fix scheduled	8
ARM			
ENGcm09831	ARM: Load and Store operations on the shared device memory regions may not complete in program order	No fix scheduled	9
ENGcm11132	ARM: A RAW hazard on certain CP15 registers can result in a stale register read	No fix scheduled	11
ENGcm11141	ARM: ARPROT[0] is incorrectly set to indicate a USER transaction for memory accesses generated from user tablewalks	No fix scheduled	13
ENGcm11135	ARM: C15 Cache Selection Register (CSSELR) is not banked	No fix scheduled	15
ENGcm11143	ARM: Cache clean memory ops generated by the Preload Engine or Clean by MVA to PoC instructions may corrupt the memory	No fix scheduled	16
ENGcm11145	ARM: Under a specific set of conditions, a cache maintenance operation performed by MVA can result in memory corruption	No fix scheduled	18
ENGcm11142	ARM: Clean and Clean/Invalidate maintenance ops by MVA to PoC may not push data to external memory	No fix scheduled	20
ENGcm11146	ARM: Incorrect L2 cache eviction can occur when L2 is configured as an inner cache	No fix scheduled	22
ENGcm11144	ARM: Swap instruction, preload instruction, and instruction fetch request can interact and cause deadlock	No fix scheduled	23
ENGcm11133	ARM: NEON load data can be incorrectly forwarded to a subsequent request	No fix scheduled	25
ENGcm11134	ARM: Under a specific set of conditions, processor deadlock can occur when L2 cache is servicing write allocate memory	No fix scheduled	27
ENGcm11212	ARM: Clarification regarding the ALP bits in AMC register	No fix scheduled -Clarified in RM	29
ENGcm10696	ARM: If a Perf Counter OVFL occurs simultaneously with an update to a CP14 or CP15 register, the OVFL status can be lost	No fix scheduled	30
ENGcm10729	ARM: A Neon store to device memory can result in dropping a previous store	No fix scheduled	32
ENGcm10707	ARM: BTB invalidate by MVA operations do not work as intended when the IBE bit is enabled	No fix scheduled	34
ENGcm10730	ARM: Taking a watchpoint is incorrectly prioritized over a precise data abort if both occur simultaneously on the same address	No fix scheduled	36
ENGcm10731	ARM: VCVT.f32.u32 can return wrong result for the input 0xFFFF_FF01 in one specific configuration of the floating point unit	No fix scheduled	38

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm11206	ARM: Cache maintenance operations by MVA for a non-cacheable memory region can result in processor deadlock	No fix scheduled	40
ENGcm11413	ARM: A Neon load from device memory type can result in data abort	No fix scheduled	42
CCM			
ENGcm11209	CCM: ARM clock source switch limitation	No fix scheduled	43
ENGcm11479	CCM: System hangs when EMI int1 clock is disabled	No fix scheduled	44
CSPI			
ENGcm11154	CSPI: Incorrectly clears the overrun status bit	No fix scheduled	45
DPLL			
ENGcm11152	DPLL: TOG_SEL bit not cleared after the TOG_DIS bit is set	No fix scheduled	46
eCSPI			
ENGcm10189	eCSPI Burst completion by SSB signal in Slave mode is not functional	No fix scheduled	47
EIM			
ENGcm12379	EIM: AUS mode is non functional for devices larger than 32MB	No fix scheduled	114
EMI			
ENGcm11038	EMI2.5: Read from M4IF Watermark status registers may have wrong result	No fix scheduled	48
ENGcm11245	EMI: WEIM 8-bit memory devices support clarification	No fix scheduled -Clarified in RM	49
ENGcm11786	EMI: EIM BCLK in DEBUG mode is not functional when GBCD is zero and BCD is not zero	No fix scheduled	50
EPIT			
ENGcm11028	EPIT: Possibility of additional pulse on src_clk when switching between clock sources	No fix scheduled	51
ESAI			
ENGcm12385	ESAI: Channel misalignment may happen in ESAI transmitter data stream when TIEN bit is zero	No fix scheduled	115
eSDCTL			
ENGcm11052	eSDCTL: Auto power down issues	No fix scheduled	52
ENGcm12377	ESDCTLv2 might fail to wait the minimal time between DDR clk & clk enable	No fix scheduled	113
eSDHC			
ENGcm11213	eSDHC: Glitch is generated on card clock with software reset or clock divider change	No fix scheduled - Clarified in the RM	53

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm11116	eSDHCv2/eSDHCv3: ADMA transfer error when the block size is not a multiple of four	No fix scheduled	54
ENGcm11115	eSDHCv2/eSDHCv3: Problem when ADMA2 last descriptor is LINK or NOP	No fix scheduled	56
ENGcm11186	eSDHCv2/eSDHCv3: eSDHC misses SDIO interrupt when CINT is disabled	No fix scheduled	57
ENGcm11406	eSDHCv3: DLL_STS_REF_LOCK status bit does not indicate when DLL is locked	No fix scheduled	58
ENGcm12290	eSDHCv3 on eSDHC3 port has setup timing issue in SD SDR mode	No fix scheduled	109
ENGcm12360	eSDHC AutoCMD12 and R1b polling problem	No fix scheduled	110
ETB			
ENGcm12386	ETB may not function properly at 1.2GHz	No fix scheduled	116
FEC			
ENGcm11214	FEC: Fast Ethernet Controller (FEC) accesses to NAND Flash Controller (NFC) does not work	No fix scheduled- Clarified in RM	59
GPT			
ENGcm11029	GPT: Possibility of additional pulse on src_clk when switching between clock sources	No fix scheduled	60
LDO			
ERR007080	LDO: On-chip LDO regulators may not enable or have a delayed output on power up	Fix planned—contact your Freescale representative	105
M4IF			
ENGcm11127	M4IF: Step-by-step mechanism violates AXI protocol	No fix scheduled	62
ENGcm10682 ENGcm10940	M4IF power-saving mode should not be enabled before DDR is configured	No fix scheduled - Clarified in RM	63
ENGcm11203	M4IF: Reading M4IF status registers of an inactive AXI master or slave stalls entire system	No fix scheduled - Clarified in RM	64
NFC			
ENGcm11215	NFC: 8-Sym ECC mode does not work with 512 byte page x16 bus NAND Flash	Feature not supported - Clarified in RM	66
ENGcm10288	NFC: Copy back function destination address restriction	No fix scheduled - Clarified in RM	67
ENGcm11124	NFC: Block write-protect does not support lock-tight	No fix scheduled	68
ENGcm11217	NFC: Block write-protect does not work in automatic operations	No fix scheduled - Clarified in RM	69
ENGcm11182	NFC: Copy-back does not work properly when addr_op = 01	No fix scheduled	70

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm11126	NFC: Software reset does not work properly under certain conditions	No fix scheduled	71
ENGcm11218	NFC: Status read does not occur at the end of the program, with RBB_MODE = 1	No fix scheduled - Clarified in RM	72
ENGcm11219	NFC: Misses read data when working in Symmetric mode with clock ratio 1:2, and using a 16-bit Flash bus width	No fix scheduled - Clarified in RM	73
ENGcm11220	NFC: Cannot reach entire address space when addr_op = 1 or 3	No fix scheduled - Clarified in RM	74
ENGcm11221	NFC: ECC mechanism may fail to report uncorrectable error situation	No fix scheduled - Clarified in RM	75
ENGcm11002	NFC can miss the sampling of the ready/busy signal (R/B) when RBB_MODE = 1	No fix scheduled	76
ENGcm11053	SDMA multi-page read from the NFC, when the WEIM is operating, can result in data corruption or EMI hanging	No fix scheduled	77
ENGcm12363	NFC wrong indication of ECC uncorrectable error occurrence after reading the spare area	No fix scheduled	112
Power Supply			
ENGcm11180	Grounding nonfunctional UHVIO IO pads power rails can cause malfunction in other UHVIO IO cells	No fix scheduled	78
ENGcm11642	UHVIO pads automatic supply voltage level detect function may not work for voltage between 1.95 V to 3.1 V	No fix scheduled	79
ROM (Boot Code)			
ENGcm11851	ROM (Boot): Boot from SATA fails when internal clock mode is used	No fix scheduled.	80
RTIC			
ENGcm10971	RTICv3 memory region unlock feature can cause the RTICv3 to hang	No fix scheduled	81
SAHARA			
ENGcm10363	SAHARA/CCM: Frequency ratio restriction for AHB and IP buses in SAHARA	No fix scheduled - Sahara spec updated	82
SATA			
ENGcm10417	SATA: IS.INFS bit is not set when ERR_C is set	No fix scheduled	83
ENGcm10975	SATA: Unknown FIS with incorrect PMP field received	No fix scheduled	84
ENGcm10983	SATA: Erroneous detection of Read Overflow condition	No fix scheduled	85
ENGcm10982	SATA: Soft Reset not sent when issued by the software during on-going transfer	No fix scheduled	86
ENGcm11018	SATA: Problems with phy_reset when Staggered Spin-Up supported	No fix scheduled	87
ENGcm11084-1	SATA: PMP field checked when CMD.PMA = 0	No fix scheduled	88
ENGcm11084-2	SATA: Data FIS received when CMD.ST = 0	No fix scheduled	89

Table 3. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ENGcm11084-3	SATA: Supported AHCI version is incorrect	No fix scheduled	90
ENGcm11084-4	SATA: DMA Setup FIS with inactive slot reception	No fix scheduled	91
ENGcm11084-5	SATA: Host does not detect disconnect in L_TMPartial/L_TMSlumber states	No fix scheduled	92
ENGcm11084-6	SATA: Possible core lockup after switching from the low power mode	No fix scheduled	93
ENGcm11084-7	SATA: Incorrect interpretation of A-bit in BIST Activate FIS	No fix scheduled	94
ENGcm11084-8	SATA: BIST responder re-timed loopback mode may drop data due to FIFO overflow	No fix scheduled	95
ENGcm11084-9	SATA: Host may not wake up from low power mode	No fix scheduled	96
ENGcm11084-10	SATA: Injected primitive error inside FIS causes CRC error	No fix scheduled	97
ENGcm11084-11	SATA: SYNC in Rx Data FIS causes PDMA FSM lock up	No fix scheduled	98
ENGcm11084-12	SATA: Synchronizer misses last wake pulse when CMD.ST bit is cleared in aggressive power mode	No fix scheduled	99
SCC			
ENGcm12354	SCC key fusing with a non-unique value	No fix scheduled	111
SRTC			
ENGcm12374	SRTC: SRTC loses data while powering up/down.	Silicon revision planned	115
SSI			
ENGcm11129	SSI: In AC97, 16-bit mode, received data is shifted by four bit locations	No fix scheduled	102
USB			
ENGcm11151	USB: Erroneous descriptor handling by USBOH module	No fix scheduled	103
ENGcm11331	USB: High Speed Transceiverless Logic interface (HS-TLL) and Full Speed Transceiverless Logic interface (FS-TLL) USB interfaces are not supported	No fix scheduled - Feature removed from the RM	104
ERR006308	USB: Host controller lock-up issue	No fix scheduled	105
VPU			
ENGcm10380	VPU: JPEG decoder does not support different AC/DC Huffman tables for Cb and Cr	No fix scheduled	106
ENGcm11195	VPU may miss generating encoding/decoding interrupt when automatic clock gating is activated	No fix scheduled	107
ENGcm11856	IPU and VPU may present some artifacts when running at 200MHz	Fixed in revision 2.1	108

ENGcm11136 AIPS: Unaligned access causes abort on writes to the internal registers

Description:

Unaligned access to AIPS can be driven high by SAHARA, DAP, and FEC. If they access the AIPS internal registers during an unaligned access, an ABORT occurs.

Projected Impact:

Unaligned accesses to the AIPS internal registers fail.

Workarounds:

Make only aligned accesses to the AIPS internal registers.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround required. Linux BSP does not use unaligned access to the AIPS registers.

WinCE BSP Status:

No software workaround required. WinCE BSP does not use unaligned access to the AIPS registers.

ENGcm09831 ARM: Load and Store operations on the shared device memory regions may not complete in program order

Description:

If a sequence of load and store operations are performed to different address locations in a memory region that is marked as shared device, then a load can incorrectly bypass a store.

The issue is reported by ARM, erratum ID 709718, Category 2[1].

Projected Impact:

If the load address and store address are mapped to access the memory region of the same device, and the device relies on memory operations to occur in program order, then this device may not operate as intended.

Workarounds:

The erratum occurs only for the shared device memory regions and not for the non-shared device memory regions. Therefore, this problem can be worked around by using the remap registers to remap all the shared device transactions to the non-shared device. The only difference between the shared device and the non-shared device is the attributes produced for the transaction on the AXI interface. Therefore, the user does not experience any impact in terms of performance from this workaround.

Another possible use of the TEX remap is to map the shared device regions to the strongly ordered transactions. This second remapping option is less desirable as it affects the performance, as strongly ordered transactions are not buffered.

The following code sequence is required to setup and enable the TEX remap. This should be done before enabling the MMU.

```
; Setup PRRR so device is always mapped to non-shared
    MRC p15, 0, r0, c10, c2, 0; Read Primary Region Remap Register
    BIC r0,#3<<16
    MCR p15, 0, r0, c10, c2, 0; Write Primary Region Remap Register

; Enable TEX remap
    MRC p15, 0, r0, c1, c0, 0; Read Control Register
    ORR r0,r0,#1<<28
    MCR p15, 0, r0, c1, c0, 0; Write Control Register
```

Another valid workaround is to place a data memory barrier (DMB) between all the memory accesses to the device regions, where ordering is required between a store and a subsequent load to a different physical address.

Proposed Solution:

No fix scheduled

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Linux BSP Status:

Fixed in Linux BSP release 10.03.00.

Shared Device memory type is not used. But PRRR setup codes were added before enabling MMU in the bootloader.

WinCE BSP Status:

Fixed in WinCE BSP release ER10_SP1.

Workaround implemented.

ENGcm11132 ARM: A RAW hazard on certain CP15 registers can result in a stale register read

Description:

Under certain conditions, a sequence of instructions where an MCR instruction that writes a CP15 register is closely followed by an MRC that reads the same register, are executed such that the RAW hazard is not detected and the MRC reads the old value of the register. This scenario can only occur for accesses to one of the following four CP15 registers:

- CacheSizeSelection Register
- Thread and ProcessID user read/write
- Thread and ProcessID user read only
- Thread and ProcessID privilege only

These registers are both readable and writable and have been optimized to execute in a single cycle.

Furthermore, this scenario occurs only when a specific sequence of instructions is executed between the MCR and the MRC. The sequence must meet two criteria:

- It must take less than three cycles to execute
- It must have one of the instructions in the following list:
  - ARM PLD with [Rn, -Rm, <shift>] addressing mode
  - ARM or Thumb PLD with [Rn, Rm,<shift>] addressing mode (unless it is LSL #0 or LSL #2)
  - Thumb or ThumbEE load/store instruction with [Rn, Rm,<shift>] addressing mode (unless it is LSL #0 or LSL #2)
  - Thumb TBB instruction

The issue is reported by ARM, erratum ID 588115, Category 3[1].

Projected Impact:

If this erratum is encountered, the old stale value of the register is read rather than the newly written value, in which case the system software may appear to behave incorrectly. However, the usage model for such a software sequence is unclear, and hence the likelihood of encountering it in practice is very low, especially considering the requirement of the second unrelated instruction that must also fall between the MCR and the MRC.

Workarounds:

If a workaround for this erratum is desired, there are two options. The first simple option is to add a NOP immediately following the MCR register write in any case where encountering this erratum may be a concern. By adding a single NOP, the minimum required cycle window is guaranteed and the erratum does not occur.

The second option is to set bit 16 in the CP15 Auxiliary Control Register. This causes a pipeline flush on every write to the CP15 register and ensures that the RAW hazard condition does not

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

occur. The second workaround has the advantage of requiring just one change to the CPU configuration that can be done statically. The disadvantage is that it has some impact on the performance of write updates to CP15 registers that would not otherwise require a pipeline flush. This second workaround can be implemented using the following code sequence to be executed in the Secure state:

```
MRC p15, 0, R1, c1, c0, 1           ; read Aux Ctl Register
ORR R1, R1 #(1 << 16)               ; set bit 16 to 1
MCR p15, 0, R1, c1, c0, 1           ; write Aux Ctl Register
```

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because PLD instruction and Thumb mode are not used for affected registers.

WinCE BSP Status:

Fixed in WinCE BSP release ER10_SP1.

BSP implements software workaround to enable pipeline flush (set bit 16 of Aux Control Register) on writes to CP15 registers.

OSBench indicates a performance impact of up to 14% for a subset of the OSBench tests (interprocess PSL calls). An analysis of the OSBench performance on Cortex-A8 determined that interprocess PSL calls generate excessive cache maintenance.

ENGcm11141 ARM: ARPROT[0] is incorrectly set to indicate a USER transaction for memory accesses generated from user tablewalks

Description:

All memory transactions performed as part of a tablewalk should be considered Privileged, even in the User mode. However, Cortex-A8 incorrectly marks memory transactions generated from tablewalks performed in User mode as the user transactions on the AXI bus. This indication is given by the ARPROT[0] signal, which is set to zero during the transaction.

The conditions are as follows:

- Cortex-A8 must be in user mode
- A memory transaction (instruction or data) misses in the TLB and causes a tablewalk
- The address for the page table entry is not found in the L2 cache, resulting in an external memory request
- This erratum occurs when APROT[0] incorrectly indicates a user transaction for this memory request on the AXI bus.

The issue is reported by ARM, erratum ID 488063.

Projected Impact:

As the values broadcast on ARPROT[0] are completely transparent to the software, the implications for this erratum are only on a specific subset of the processor systems, specifically for a system that includes some form of system level memory protection unit, that uses the ARPROT bits to determine if a memory request can be allowed. For any system that does include such a unit, that unit may report false errors on page table accesses due to this erratum.

Workarounds:

As the processor directly does not make use of ARPROT[0], any workaround would be specific to the device that makes use of the values broadcast on ARPROT[0]. The most likely usage would be some form of system memory protection unit. If such protection unit exists, it may need to filter out any access to the page tables from the address space that is protected to operate properly. This implies that the external protection unit cannot provide additional protection for the page tables. For example, the page table cannot be inserted in a Secure RAM which cannot be accessed in User mode, as in this case, an additional protection is added beside the MMU. Alternatively, the CSU can be configured to transform User access to Privileged on addresses used by PAGE TABLE.

Proposed Solution:

No fix scheduled

Linux BSP Status:

System memory bus has no user mode protection, so this is not applicable.

WinCE BSP Status:

No software workaround is available.

ENGcm11135 ARM: C15 Cache Selection Register (CSSELR) is not banked

Description:

ARMv7 architecture specifies that the CSSELR should be banked between Secure and Non-secure states. Cortex-A8 does not currently bank this register.

The conditions are as follows:

- The system should have an active process in secure state and an active process in non-secure state at the same time.
- The system should perform cache maintenance operations in both secure and non-secure processes.

The issue is reported by ARM, erratum ID 485963, Category 2[1].

Projected Impact:

A cache cleaning sequence that reads the CSSELR may not work as expected. The published sequence for cleaning the entire cache (see *ARM Architecture Reference Manual*) includes setting the CSSELR followed by a read from the selected Cache Size ID register (CCSIDR). If the non-secure side executes this sequence, and is encountered by a secure interrupt between the setting of the CSSELR and the reading of the selected CCSIDR, then there is a possibility that the secure side may also use the CSSELR. On returning to the non-secure side, the CSSELR value may have changed, which makes the cache cleaning sequence to malfunction. Similarly, a non-secure interrupt can cause a secure cache cleaning sequence to malfunction.

Workarounds:

When transitioning security state, the secure monitor software should save the current CSSELR value (corresponding to the security state the processor is transitioning out of) and restore the previously saved CSSELR value (corresponding to the security state the processor is transitioning into).

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround required. Trustzone is not used.

WinCE BSP Status:

No software workaround required. System does not perform cache maintenance in non-secure mode.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

ENGcm11143 ARM: Cache clean memory ops generated by the Preload Engine or Clean by MVA to PoC instructions may corrupt the memory

Description:

When a Clean to Point of Coherency (PoC) by MVA operation is performed, or the Preload Engine is programmed to clean a region of memory from the L2 cache, a cache line from that region can be corrupted with a stale copy of memory, and a memory store operation is lost.

The conditions are as follows:

- A Cache Clean by MVA to the PoC instruction is executed to clean cache line A, or the preload engine is configured to clean a memory region which includes cache line A. Either of the operations result in the placement of cache line A into a victim buffer for writeback to external memory. It also keeps the line still valid in the L2 cache.
- A memory store operation is performed to the same cache line A that is evicted by the cache clean operation. This operation results in a modification of cache line A in the L2 cache (but not to the copy of the line that may still remain in the victim buffer if memory response is slow).
- A cache eviction is done of cache line A due to an unrelated memory request to load cache line B. The modified copy of cache line A is placed in a victim buffer. At this point, the two victim buffers may contain two different versions of cache line A. As each victim buffer uses a different AXI ID and arbitrates independently for the AXI bus, there is no guarantee for the order in which the memory updates occur, and the store operation may be overwritten by the cache clean operation, leaving the external memory with stale contents.

The issue is reported by ARM, erratum ID 586323, Category 2[1].

Projected Impact:

If the operation sequence occurs as described above, one or more store operations are lost, resulting in incorrect program behavior. This can occur for any application which either uses the preload engine to clean a memory region, or uses Clean by MVA to PoC maintenance operations to clean a region of memory.

Workarounds:

There are two feasible workarounds that can be used for this erratum. The first workaround is to place a DMB or DSB barrier at the end of each cache clean routine or preload engine memory clean sequence. This barrier operation ensures that the cleaned line goes out and is seen by main memory before the store is executed and therefore guarantees that the clean is done correctly and memory contains the correct final value.

This workaround is consistent with the ARM recommended practice for ending the maintenance routine. The above workaround is convenient to implement and should work for all expected usage models. However, there is still the possibility that an interrupt can be taken during the clean routine, and the interrupt handler can perform a store operation to the line just cleaned, allowing for the scenario which can lead to the erratum.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Another workaround that avoids even the case mentioned above, is to convert all Clean by MVA to PoC operations to Clean and Invalidate by MVA to PoC as described in the code sequence as follows:

- Replace all uses of: MCR p15, 0, <Rn>, c7, c10, 1;
- Clean Data cache line by MVA to PoC with this instruction: MCR p15, 0, <Rn>, c7, c14, 1;
- Clean and Invalidate cache line by MVA to PoC.

There is no Preload Engine equivalent for the second workaround option as it is not possible to configure the preload engine to perform a clean and invalidate operation. Therefore, if there are concerns that the DSB based workaround is insufficient, then it is advisable to not use the Preload Engine for cleaning memory regions. The preload engine can be configured such that it is not accessible at user/privilege and nonsecure/secure level of granularity.

For more information on Preload Engine configurability, see *Cortex-A8 Technical Reference Manual*.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

All cache flush routines have DSB at end.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER10_SP1.

Interrupts are enabled during cache clean operation, so DMB/DSB workaround is not sufficient.WinCE BSP implements software workaround to replace clean cache line operation with Clean-and-Invalidate cache line operation. There may be a performance penalty due to the undesired invalidation of the cache line when invoking OEMCacheRangeFlush to clean individual cache lines.

ENGcm11145 ARM: Under a specific set of conditions, a cache maintenance operation performed by MVA can result in memory corruption

Description:

If a non-cacheable memory request is subsequently followed by any cache maintenance operation done by MVA, then the memory can be corrupted.

The conditions are as follows:

- The L1 data cache must be of size 32 Kbyte
- The L1 data cache hardware alias checks are enabled (the L1ALIAS bit in the Auxiliary Control Register is set to 0)
- The virtual memory management used by the operating system does not follow the page coloring guidelines and allows virtual to physical address alias cases to exist on bit 12 of the address
- A non-cacheable memory request to normal, device, or strongly ordered memory is subsequently followed by a cache maintenance operation done by MVA without any cacheable memory operations executed in between. The non-cacheable memory request can be fully executed, or can be a speculative instruction in the branch shadow that subsequently is flushed.

When the above conditions are met and the cache maintenance operation is performed to generate a hash alias scenario on its cache lookup, memory corruption or a false parity error can occur.

The issue is reported by ARM, erratum ID 586324, Category 2[1].

Projected Impact:

If the operation sequence occurs as described above, then memory can be corrupted or a false parity error can be generated. In addition, even if the workaround as described below is implemented, it is possible that a nonsecure maintenance operation could result in the invalidation of a secure memory location. Therefore, this could possibly be viewed as an avenue for a security attack. However, the contents of secure memory cannot be viewed as a direct result of this erratum and the lack of consistent repeatability makes it very difficult for the user to make use of this erratum as a security attack.

Workarounds:

If full PIPT caching support is not required by the operating system, or the processor includes a 16 Kbyte L1 data cache, then no workaround is required. If alias conditions can occur, then the workaround is to guarantee that a cache maintenance operation is not immediately preceded by a non-cacheable memory request. This is guaranteed by initiating every cache maintenance by MVA routine with a cacheable load or store request immediately preceding the main loop and ending with a DSB barrier operation at the end of the loop. The load or store that precedes the loop can be done to any cacheable memory location. In addition, both interrupts and aborts should be masked during the cache maintenance routine. Interrupt masking is required to prevent a non-cacheable memory request, either fully executed or in a branch shadow, from initiating the sequence that can

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

result in this erratum. If there are concerns about the interrupt latency, the maintenance loop can be amended to enable and disable the interrupts directly around the maintenance operation. This impacts the time taken to complete the maintenance loop.

To workaround any concerns of a potential security attack due to this erratum, all secure memory should be marked as inner write through. This can be done either by using the caching attributes in the page tables for all secure page tables or by making use of the secure banked version of the remap registers. Apart from making all secure memory write through, a routine should be run out of reset to completely fill the cache with dummy data, to prevent invalid, uninitialized data in the cache from being written out to memory and potentially corrupting secure memory. Making all secure memory inner write through guarantees that even if the invalidation of a secure line in the L1 cache occurs due to this erratum, the correct data is not lost.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Does not apply to Linux BSP because page coloring guidelines are followed for VIPT cache types.

WinCE BSP Status:

It was determined that the conditions required by the erratum do not occur within WinCE, and therefore, no software workaround is required.

ENGcm11142 ARM: Clean and Clean/Invalidate maintenance ops by MVA to PoC may not push data to external memory

Description:

When a Clean to Point of Coherency or Clean and Invalidate to Point of Coherency by MVA operation is performed, it is possible that the line remains present in the L2 cache and any dirty data is not pushed out on to the AXI bus to main memory. This can occur whenever the requested address is present in the L1 cache but not the L2 cache.

The conditions are as follows:

- The memory region being cleaned is configured in write allocate mode
- The cache line being cleaned is initially present in the L1 cache and not in the L2 cache

The issue is reported by ARM, erratum ID 586320, Category 2[1].

Projected Impact:

If a Clean or Clean and Invalidate operation does not operate as intended, and leaves the data present in the L2 cache, the memory coherency in the system can no longer be guaranteed. Therefore, this erratum impacts any code sequence used to maintain the system coherence.

Workarounds:

The software workaround for this erratum is to disable the write allocate in the L2 cache, as shown in the following instruction sequence:

```
MRC p15, 1, <Rd>, c9, c0, 2; read L2 cache Aux Ctrl Reg
ORR <Rd>, <Rd>, #(1 << 22); set the Write Allocate disable bit
MCR p15, 1, <Rd>, c9, c0, 2; write the L2 cache Aux Ctrl Reg
```

Disabling the write allocate in the L2 cache can impact the performance of some applications. If this performance impact is deemed to be very high, there are two other software workarounds that can be used. The first is to disable write allocate around each sequence of clean by MVA to PoC or clean/invalidate by MVA to PoC instructions, as shown in the following instruction sequence:

```
MRC p15, 1, <Rd>, c9, c0, 2; read L2 cache Aux Ctrl Reg
ORR <Rd>, <Rd>, #(1 << 22); set the Write Allocate disable bit
MCR p15, 1, <Rd>, c9, c0, 2; write the L2 cache Aux Ctrl Reg

<perform sequence of MVA operations here>

MRC p15, 1, <Rd>, c9, c0, 2; read L2 cache Aux Ctrl Reg
BIC <Rd>, <Rd>, #(1 << 22); clear the Write Allocate disable bit
MCR p15, 1, <Rd>, c9, c0, 2; write the L2 cache Aux Ctrl Reg
```

The final workaround that can be implemented is to perform each maintenance operation twice with interrupts disabled. By performing the operation twice in back-to-back successions with no other memory operations executed in between, it can be assured that the line is evicted from both L1 and L2 cache and written out to main memory.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Perform the following steps:

1. Disable the interrupts and the imprecise aborts
2. Execute the maintenance operation first pass
3. Execute the same maintenance operation, second pass
4. Enable the interrupts and the imprecise aborts

Repeat the above sequence for each cache maintenance operation. Interrupts can remain disabled for a longer sequence of maintenance operations, but this has a negative effect on interrupt latency. This workaround has a performance impact on the execution time of cache maintenance operations.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

The software workaround is to disable write allocate in the Level 2 cache in the bootloader. This workaround has performance penalty.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER1.

Write allocate is disabled in L2 cache control register. This workaround impacts performance.

ENGcm11146 ARM: Incorrect L2 cache eviction can occur when L2 is configured as an inner cache

Description:

Under specific set of conditions, the stale data saved in the L2 cache can be erroneously returned to the processor on a subsequent load instruction.

The conditions are as follows:

- The L2 cache must be configured as an inner cache rather than as an outer cache
- The L2 cache must be configured to use write allocate memory type

The issue is reported by ARM, erratum ID 468413, Category 2[1].

Projected Impact:

If this erratum occurs, stale data can be read by a subsequent load instruction, resulting in an incorrect program behavior.

Workarounds:

There are two viable workarounds for this erratum. One workaround is, not to configure the L2 cache as an inner cache, but maintain the default setting as an outer cache. The second workaround is to use the remap registers to remap the inner cache attributes from write allocate to write back instead.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

The software workaround is to disable write allocate in the Level 2 cache in the bootloader. So no condition is to trigger this issue. This workaround has performance penalty.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER1.

Write allocate is disabled in L2 cache control register.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

ENGcm11144 ARM: Swap instruction, preload instruction, and instruction fetch request can interact and cause deadlock

Description:

Three memory requests in the L2 cache can interact and result in a deadlock condition. The exact scenario involves a dependency chain of three requests, an instruction fetch request, a memory preload instruction (PLD) and a swap instruction (SWP). In this dependency loop, no request can progress as each one of them is dependent on the next request. That is, the PLD request cannot complete as the IF request is pending to use the BIU. The IF request cannot complete because of the pending SWP request, and the SWP request is not allowed to complete as it is waiting on the PLD to complete before obtaining the lock on the bus.

The conditions are as follows:

- PLD instructions must be used by the processor
- SWP instructions must be used by the processor

The issue is reported by ARM, erratum ID 468415, Category 3[1].

Projected Impact:

This erratum only impacts the users of swap instructions. Swap instructions have been deprecated from the ARMv7 version of the ARM Architecture as its functional use in terms of setting up semaphores is now replaced from the ARMv6 architecture forwards by the LDREX and STREX instructions. If this erratum is encountered and the processor deadlock occurs, it can only be interrupted by resetting the processor.

Workarounds:

One software workaround for this erratum is, not to use the swap instructions. If swap instructions are to be used in the code base, the other software workaround is to disable the PLD instructions and make them a NOP. The code required to implement this workaround is as follows:

```
MRC p15, 0, r0, c1, c0, 1; read register
ORR r0, r0, #(1<<9); PLDNOP - force PLD to be NOP
MCR p15, 0, r0, c1, c0, 1; write register
```

This workaround has some performance impact on the peak memory copy bandwidth.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not implemented because the swap instructions are not used. They are deprecated for ARMv7.

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

WinCE BSP Status:

Kernel Interlocked APIs do not make use of swap instructions. No usage of swap instruction found in the public/private code that Freescale has access to. Confirmed that swap instruction is not used in the WinCE OS.

Freescale codecs and customer application code must avoid usage of swap instructions.

ENGcm11133 ARM: NEON load data can be incorrectly forwarded to a subsequent request

Description:

Under very specific set of conditions, data from a Neon load request can be incorrectly forwarded to a subsequent, unrelated memory request.

The conditions are as follows:

- Neon loads and stores must be in use
- Neon L1 caching must be disabled
- Trustzone must be configured and in use
- The secure memory address space and the non-secure memory address space both use the same physical addresses, either as an alias or the same memory location or for separate memory locations

The issue is reported by ARM, erratum ID 468414, Category 2[1].

Projected Impact:

If this erratum is encountered, it is possible for a load request to receive the wrong data value which can likely result in incorrect operation of the program.

Workarounds:

There are many software solutions for this erratum and only one has to be applied. The recommended solution, if possible, is to map cacheable areas of memory so that both secure and non-secure do not share the same physical address space.

Another possible solution is to force NEON to cache in the L1 data cache. This can be programmed using the Auxiliary Control Register bit [5], L1NEON, as follows:

```
MRC p15, 0, r0, c1, c0,             1; read register
ORR r0, r0, #(1<<5)                 ; L1NEON caching enable
MCR p15, 0, r0, c1, c0, 1           ; write register.
```

Another possible solution is to disable L2 data forwarding from the victim buffers. This can be programmed using the L2 Auxiliary Control Register bit[27], Load data forwarding disable as follows:

```
MRC p15, 1, r0, c9, c0, 2           ; read register
ORR r0, r0, #(1<<27)                ; L2 load data forwarding disable
MCR p15, 1, r0, c9, c0, 2           ; write register
```

Both workarounds can be implemented with little or no perceived performance impact in the majority of applications.

Proposed Solution:

No fix scheduled

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

Linux BSP Status:

Trustzone is not used and Neon L1 caching is enabled. So this case does not occur.

WinCE BSP Status:

Trustzone is not used in WinCE BSP. Therefore, the erratum does not apply.

The issue occurs when two physical addresses in both Secure (S) and Non-Secure (NS) worlds are required. If the use of the NS world is never enabled, then the issue cannot occur. The hazard occurs between the NS and S worlds which does not get flushed properly, and data gets forwarded from the L2 data buffers.

ENGcm11134 ARM: Under a specific set of conditions, processor deadlock can occur when L2 cache is servicing write allocate memory

Description:

If a load request is processed which misses the L2 cache, but cannot be immediately forwarded to the BIU, it encounters a special hazard which prevents the request from being required to access the L2 cache RAM again to save power. There can be multiple requestors with unique addresses, (that is, one address per cache line) with this special hazard. All write-allocate requests that access the L2 cache RAM, on port1, do not have address comparators to check for this special hazard condition. So, if a subsequent write-allocate request is issued to the L2 cache RAM on port1 and allocates a victim buffer, then all requests pending with this special hazard must be forced to perform a L2 cache RAM lookup again to maintain memory coherency. There is a 1-cycle window in which the write-allocate request must allocate to a victim buffer and a pending request to the BIU is not prohibited from going to the BIU, such that a deadlock can occur.

The conditions are as follows:

- The processor must have L2 cache present and enabled.
- The L2 cache must be configured to support the write allocate memory type.

The issue is reported by ARM, erratum ID 468416, Category 2[1].

Projected Impact:

If this erratum is encountered and processor deadlock occurs, it can only be interrupted by asserting RESET on the processor.

Workarounds:

The workaround for this erratum is to disable write-allocate by programming the L2 Auxiliary Control Register bit[22], Write allocate disable:

```
MRC p15, 1, r0, c9, c0, 2; read register
ORR r0, r0, #(1&lt;&lt;22); Write allocate disable
MCR p15, 1, r0, c9, c0, 2; write register
```

Disabling write allocate in the L2 cache could have a performance impact for some applications.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

The software workaround is to disable write allocate in the Level 2 cache. This workaround has performance penalty.

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER1.

Write allocate is disabled in L2 cache control register.

ENGcm11212 ARM: Clarification regarding the ALP bits in AMC register

Description:

In the register AMC of the Tiger/ARM Platform (0xBASE_0018) the bit ALPEN must be set to 1 and ALP[2:0] must be set to ‘000’. Other combinations are reserved and must be avoided.

Projected Impact:

Memory retention issues unless the guideline is followed.

Workarounds:

None

Proposed Solution:

No fix is scheduled. A clarification will be added to the reference manual.

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP release SDK_1.6.

In the register AMC of the ARM Platform (0xBASE_0018), the bit ALPEN must be set to 1 and ALP[2:0] must be set to ‘000’. Other combinations are reserved and must be avoided.

ENGcm10696 ARM: If a Perf Counter OVFL occurs simultaneously with an update to a CP14 or CP15 register, the OVFL status can be lost

Description:

If the PMU is in use and an overflow event occurs simultaneously with a write to one of the subsets of CP15 and CP14 registers, the overflow event can be lost.

The conditions are as follows:

1. The performance counters must be in use
2. The performance counter must have an overflow (counter value goes beyond 0xFFFF_FFFF)
3. Simultaneous with the counter overflow, a MCR instruction must be executed that writes to one of the following CP14/CP15 registers:
   - Any PMU register other than PMU counter registers
   - ThumbEE Configuration Register
   - ThumbEE Handler Base Register
   - System Control Register
   - Auxiliary Control Register
   - Secure Configuration Register
   - Secure Debug Enable Register
   - Nonsecure Access Control Register
   - Context ID and Thread ID Registers
   - Coprocessor Access Register
   - Cache Size Select Register

The issue is reported by ARM, erratum ID 628216, Category 2[1].

Projected Impact:

If the erratum occurs, the overflow status flag is not set for that counter in the Overflow Flag Status Register, and an interrupt request is not generated, even when the Interrupt Enable Set Register is configured to generate an interrupt on counter overflow.

Workarounds:

The main workaround is to poll the performance counter. The maximum increment in a single cycle for a given event is 2. Therefore, polling can be infrequent as no counter can increment by more than 2^{32} in fewer than 2 billion cycles.

If the main usage model for performance counters is collecting values over a long period, then polling can be used to collect values (and reset the counter) rather than waiting for an overflow to occur. Polling can be done infrequently and overflow can be avoided.

If the main usage model for performance counters relies on presetting the counter to some value and waits for an overflow to occur, then polling can be used to detect when an overflow event is

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

missed. An overflow can be determined to have been missed if the unsigned value in the counter is less than the value preset into the counter. Polling can be done infrequently because of the number of cycles it requires for this check to fail. If the erratum is triggered and an overflow event is missed, the counter sample can be thrown away or the true value can be reconstructed.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround is implemented because performance counters are not used and are only for debug.

WinCE BSP Status:

WinCE BSP currently does not use the performance counters. No BSP change required.

ENGcm10729 ARM: A Neon store to device memory can result in dropping a previous store

Description:

If a Neon store is done to Device type memory and is followed in instruction sequence by a load instruction to Device type memory, it is possible that an unrelated store instruction that is done to cacheable memory and hit the L1 cache has its data dropped and therefore not update memory.

There are three different memory types defined in the ARM architecture namely, Strongly Ordered, Device, or Normal. Device type memory is one of the three different memory types. This region is specified by the page table entries used by the MMU.

The conditions for this erratum are that relatively close in the instruction stream, the following must occur:

- A Neon store is done to Device type memory.
- A load is executed to Device type memory (any load to Device type memory region, not just from Neon), consecutive to the Neon store.
- Several stores hit the L1 cache. (Any store that hit the L1 cache - Neon or integer core. The address does not matter.)

The issue is reported by ARM, erratum ID 507113, Category 3[1].

Projected Impact:

If the erratum occurs, one or more cacheable stores that hit the L1 cache do not update the cache, leaving stale contents in memory. This is likely to cause observable, incorrect behavior in the application.

The Neon access to memory region marked as Device is not a practical case in general.

Workarounds:

The only workaround for this erratum is to avoid accessing the Device type memory with Neon store instructions. (There should be no practical case for this, anyway). However, if needed, define the region as Strongly Ordered memory, instead.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because Device memory type is not user space accessible.

WinCE BSP Status:

Workaround implemented in WinCE BSP release APR2010.

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

The memory which is Device Shared should be mapped as Normal Outer/Inner Non-Cacheable. This is the preferred memory type for RAM memory mapped as NCB. Customer software must avoid Device memory types.

ENGcm10707 ARM: BTB invalidate by MVA operations do not work as intended when the IBE bit is enabled

Description:

All BTB invalidate operations, including BTB Invalidate by MVA operations, by default are implemented as a NOP in the Cortex-A8 processor. These operations can be executed as NOPs as flushing BTB entries are not required by the Cortex-A8 processor for correct functionality, and there is no additional performance penalty for an incorrect branch prediction versus a non-prediction. However, it is possible for BTB operations to be enabled by setting the IBE bit in the CP15 Auxiliary Control Register. When enabled in this fashion, BTB invalidate by MVA operations may not work as intended. Instead of writing zeros to the valid bit of the BTB entry matching the MVA provided, the CP15 “Invalidate Branch Predictor by MVA” operation writes the value currently in the “Instruction L1 System Array Debug Register 0.” This register is not initialized at the reset time and can only be written in secure, privileged modes when CP15SDISABLE is not set.

The conditions are as follows:

1. The branch predictor is enabled (SCTLR.Z = 1)
2. The Auxiliary Control Register IBE bit is set to 1
3. An invalidate Branch predictor by MVA operation is executed
4. The Instruction L1 System Array Debug Register 0 contains a non-zero value which sets the valid bit and clears the page cross bit.

The issue is reported by ARM, erratum ID 687067, Category 3[1].

Projected Impact:

If the non-zero value contained in L1 System Array Debug Register 0 sets the valid bit of the BTB entry, then the entry is not invalidated as intended.

Workarounds:

A workaround for this erratum is, not to enable the IBE bit. ARM recommends that the IBE bit should not be enabled unless it is required for an erratum workaround.

If the IBE is to be enabled, then the L1 System Array Debug Register 0 should be initialized to a zero value. This register is for RAM array debug purposes and is not used as a part of normal functionality. It is only accessible in a privileged secure mode. Therefore, it can be statically initialized as a part of the boot code sequence. If the register is used for debug purposes, the value should be reset to zero when the debug sequence completes.

The code to initialize the L1 System Array Debug Register 0 is as follows:

```
MOV r1, #0
MCR p15, 0, r1, c15, c1, 0 ; write instruction data 0 register
MRC p15, 0, R1, c1, c0, 1 ; read Aux Ctl Register
ORR R1, R1 #(1 << 6) ; set IBE to 1
MCR p15, 0, R1, c1, c0, 1 ; write Aux Ctl Register
```

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround is implemented because IBE is not set as 1.

WinCE BSP Status:

WinCE BSP does not set the IBE bit. BTB invalidate operations will be NOPs. No BSP change required.

ENGcm10730 ARM: Taking a watchpoint is incorrectly prioritized over a precise data abort if both occur simultaneously on the same address

Description:

If a debug watchpoint and a precise data abort are both triggered from the same data access, the ARM Architecture specifies that the data abort should be prioritized. However, this does not occur on the Cortex-A8 and the watchpoint is taken instead.

The conditions for the erratum are as follows:

1. At least one debug watchpoint is programmed
2. A precise data abort occurs on the same address as the watchpoint

The issue is reported by ARM, erratum ID 693270, Category 3[1].

Projected Impact:

The implications of this erratum only affects the debug software. The data abort should take precedence over the watchpoint so that the OS has a chance to fix up paged-out memory before re-executing the instruction and presenting the debugger with the watchpointed address. Due to this erratum, this fix up does not occur and the debugger should be capable of handling a faulting address.

Workarounds:

The workaround for this erratum is to ensure that the debugger software handles the faulting address. When the debugger is signalled a watchpoint, and identifies that the page being accessed is subjected to an MMU fault, which it would like the OS to patch up before dealing with itself, it can perform the following actions:

- Disable the watchpoint
- Set vector catch on the local Data Abort exception (secure or non-secure, as appropriate)
- Set the PC at the watchpointed instruction and restart execution

The processor restarts, re-executes the instruction and generate the MMU fault. It then fetches the instruction from the Data Abort handler and re-enter Debug state because of the Vector Catch event. The debugger can then perform the following actions:

- Re-enable the watchpoint
- Disable the vector catch
- Set the PC at the Data Abort vector and restart execution

The processor restarts and re-executes the Data Abort vector instruction. The OS then patches up the MMU fault and attempts to re-execute the original instruction. Re-executing the instruction regenerates the Watchpoint debug event, but now the page is properly patched up.

Proposed Solution:

No fix scheduled

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

Linux BSP Status:

No software workaround is implemented because the watchpoints are for debug only.

WinCE BSP Status:

Watchpoints are only used by debug software. WinCE uses Microsoft kernel debugger that does not utilize hardware watchpoints. No BSP change required.

ENGcm10731 ARM: VCVT.f32.u32 can return wrong result for the input 0xFFFF_FF01 in one specific configuration of the floating point unit

Description:

If the integer to floating point conversion operation, VCVT.f32.u32, is executed with the FPSCR register configured for Default NaN and Flush-to-zero enabled, and the rounding mode used is RP (Round-to-Positive infinity), it returns the incorrect result for the source operation 0xFFFF_FF01. Specifically, it returns the result 0x0000_0000 instead of the correct result 0x4F80_0000. The erratum can occur only for this specific input value and this specific configuration of the FPSCR register.

The conditions are as follows:

1. Default NaN is enabled (FPSCR[25] = 1'b1)
2. Flush-to-zero is enabled (FPSR[24] = 1'b1)
3. RP rounding mode is enabled (FPSR[23:22] = 2'b01)
4. A VCVT.f32.u32 instruction is executed with the source operand 0xFFFF_FF01
5. The result of the instruction is incorrect 0x0000_0000 rather than 0x4F80_0000

The issue is reported by ARM, erratum ID 715847, Category 3[1].

Projected Impact:

The incorrect result from the conversion operation can result in further incorrect results calculated and unexpected program behavior.

Workarounds:

The erratum only occurs if the floating point unit is configured in run fast mode with RP rounding. The easiest workaround is to avoid using this particular mode combination. Round-to-Nearest (RN) is a common rounding mode used, but if RP functionality is desired, it should be done without using Default NaN and/or without Flush-to-zero enabled. Default NaN signalling, Flush-to-zero, and rounding mode are all configured using bits [25:22] of the FPSCR register. This register is typically configured by the system software and should not change within an application.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround is implemented because the RN mode is used.

WinCE BSP Status:

The WinCE BSP configures the VFP for run-fast mode (default NAN enabled, flush-to-zero enabled), so it is subject to the erratum. WinCE BSP does not specifically configure RP rounding

1. Category 3 defined as: Behavior that is not the originally intended behavior but should not cause any problems in applications.

mode which is another condition for this erratum. Our FP support comes from a DLL provided by ARM. The ARM DLL should avoid the specific rounding mode associated with the erratum. Need to ensure RP rounding mode is not enabled.

ENGcm11206 ARM: Cache maintenance operations by MVA for a non-cacheable memory region can result in processor deadlock

Description:

If a Clean by MVA, Invalidate by MVA, or Clean and Invalidate by MVA cache maintenance operation is performed in a memory region that is marked non-cacheable, device, or strongly ordered, it is possible for the processor to deadlock or have stale data left in the processor. This erratum occurs when the address hits the cache in a way that is not predicted by the Hash Virtual Address Buffer (HVAB), which is a cache way predictor inside the processor. This erratum can occur only for the cache maintenance operations that are performed by MVA. It does not occur for the set/way based cache maintenance operations.

The conditions are as follows:

1. A memory region is marked cacheable in a page table entry, and a cache line from that region is placed in the data cache
2. A second page table entry marks the same memory region as non-cacheable, device, or strongly ordered. This can occur by changing the memory attributes in the existing page table entry, or through an alternative page table entry that maps the same virtual to physical address but with non-cacheable, device, or strongly ordered attributes rather than cacheable
3. A Clean by MVA, Invalidate by MVA, or Clean and Invalidate by MVA cache maintenance operation is done to this address
4. The maintenance operation receives a false hit indication from the HVAB array
5. The maintenance operation receives a true hit indication from the Tag lookup, which implies that the data is present in the array, but located in a different way that is not predicted by the HVAB
6. An eviction of the dirty line has started but not finished, and the processor leaves stale data in the cache and can potentially enter a deadlock state

The issue is reported by ARM, erratum ID 728018, Category 2[1].

Projected Impact:

If stale data is left in the cache, the processor does not work as intended. If deadlock state occurs, it can only be exited by asserting the RESET pin on the processor.

Workarounds:

There are two possible workarounds for this erratum.

The first workaround is to avoid performing the cache maintenance operations to non-cacheable addresses previously marked cacheable and therefore may be resident in the cache. If the address is present in the cache, it implies that the memory region is marked cacheable at some earlier point of time and explicitly changed to non-cacheable before the maintenance operation is performed. If the region type is not changed to non-cacheable before executing the maintenance operation, this erratum can be avoided. The value of changing a memory region from cacheable to non-cacheable

1. Category 2 defined as: Behavior that contravenes the specified behavior and that can limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.

before performing the maintenance operations is that this is the only way that the ARM v7 Architecture guarantees the line is not immediately placed back into the cache due to the possibility of data speculation. However, in Cortex-A8, this degree of data speculation is never done. Therefore, changing the memory type to non-cacheable before executing the cache maintenance operation is not required to assure that the line is not immediately placed back into the cache. However, if there is a code compatibility with other v7 implementations (that may exhibit this level of data speculation) is a concern, then this first workaround is insufficient, and the second workaround should be used.

The second workaround is to execute the loop of cache maintenance operations twice. Execute the loop once with the memory region still marked cacheable. Then change the page table entry to make the memory region non-cacheable and execute the loop for a second time. The first loop cleans the data from the cache in the Cortex-A8. On the Cortex-A8, the second loop is redundant as it misses on all lines in the cache, but resolves the data speculation issue that can occur on a different v7 architecture implementation. The existing cache maintenance code in a dynamically paged environment can be dependent on the maintenance operation triggering a page fault to set the correct page table entry. The workaround code must independently ensure that the correct page table entry is present.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because set/way is used in cache maintenance.

WinCE BSP Status:

Not implemented

ENGcm11413 ARM: A Neon load from device memory type can result in data abort

Description:

When Neon performs a single byte load from Strongly Ordered or Device type memory with an access size of more than 8 bytes, the system AXI bus issues a burst access which is longer than 8 beats of a single byte.

However, the M4IF is capable of supporting only access with a burst length of less than or equal to 8, as indicated in the *i.MX53 Applications Processor Reference Manual* (MCIMX53RM). When an access with burst length greater than 8 beat is detected by the M4IF, it is not forwarded to the memory controllers. Instead, a DECERR AXI bus error is indicated and data abort exception is sent back to the master.

Conditions for this issue:

1. A single byte Neon load is issued with more than 8 bytes access, for example:
   ```
   Vld1.8 {d0,d1,d2,d3}, [Rs]!
   ```
2. The source address is Strongly Ordered or Device memory type

Projected Impact:

The Neon access to memory region marked as Strongly Ordered or Device are not usually a practical case in general. Note that there are also other reported limitations for Neon access to Device type memory such as ENGcm10729.

Workarounds:

Several software solutions can be proposed for this issue:

- Use 8-bit Neon load with access size of less or equal to 8 bytes. For example:
  ```
  Vld1.8 {d0-d1}, [Rs]!
  ```
  This solution results in some performance degradation.
- Use 16 or 32-bit Neon load instructions instead. For example:
  ```
  Vst1.32 {d0 – d3}, [Rs]!
  ```
  The limitation of this proposal is that the source data address must be 16 or 32-bit aligned.
- Define the memory region as Normal Non-Cacheable type instead of Strongly Ordered or Device memory type. In this case, need to avoid potential memory consistence issues and perform Data Synchronization Barrier (DSB) before other DMA engine access the region for read, as the Write Buffer is enabled.

Proposed Solution:

No fix scheduled

ENGcm11209 CCM: ARM clock source switch limitation

Description:

There are two muxes which select a clock source for pll1_sw_clk (which is also the source for ARM clock). One of them is a regular mux (step select mux), and the other is a synchronous mux. The clock sources are selected by a single register (CCSR). If the pll1_sw_clk_sel bit is cleared (CCSR[2]) and the selection of the regular mux (CCSR[8:7]) is changed at the same time, then the regular mux is likely to switch first and can cause a glitch on pll1_sw_clk and hence on ARM clock and possibly other clocks as well.

Due to above, the CCSR[8:7] bits may only be modified when step_clk is no longer selected. Therefore, CCSR[2] must be cleared in separate register access prior to changing CCSR[8:7].

Projected Impact:

None, if the proposed workaround is implemented.

Workarounds:

The CCSR[8:7] bits can be modified only when step_clk is no longer selected. The ARM clock source selection should be done in two accesses. The CCSR[2] must be cleared in separate register access prior to changing CCSR[8:7].

Proposed Solution:

No fix is scheduled. A clarification is added to the reference manual.

Linux BSP Status:

Partially implemented. Full implementation in next release.

WinCE BSP Status:

Implemented.

ENGcm11479 CCM: System hangs when EMI int1 clock is disabled

Description:

When the emi_int1 clock in the CCM_CCGR5 register is disabled, the transaction to the CCM passes through the INT1 channel of the EMI. However, the BRESP cannot be received by the ARM because the clocks are already turned off. This causes the system to hang.

Projected Impact:

None. User should refrain from disabling the EMI int1 clock.

Workarounds:

None

Proposed Solution:

No fix scheduled. A clarification is added to the reference manual.

ENGcm11154 CSPI: Incorrectly clears the overrun status bit

Description:

The CSPI automatically clears the overrun error status bit when the RxFIFO is read. This bit should not be cleared. This bit is designed for the interrupt access mode, and not for the DMA access mode.

The conditions are as follows:

- When the RO bit is cleared by an RxFIFO read, it does not cause a problem if no DMA accesses to the CSPI occur
- When DMA is utilized, the interrupt status of RO can be lost because of uncontrolled RxFIFO access by DMA

Projected Impact:

If the RxFIFO is read before reading the Overrun error status bit, it is possible to miss the Overrun and thus miss the data.

Workarounds:

When DMA is used for data transfers, the software can program the CSPI to only allow the interrupt generation during the overrun condition and not enable any other interrupt sources. In this way, whenever an interrupt comes from CSPI, the software can assume that it is the result of an Overrun condition.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required. DMA mode is not enabled in Linux BSP.

WinCE BSP Status:

Implemented.

ENGcm11152 DPLL: TOG_SEL bit not cleared after the TOG_DIS bit is set

Description:

Under certain conditions, the DPLL IP TOG_SEL bit is not cleared after the TOG_DIS bit is set. This issue is random in nature.

Projected Impact:

The proposed workaround resolves the issue.

Workarounds:

A software delay for a fixed amount of time based on TOG_COUNT after the TOG_DIS bit is set.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because not used

WinCE BSP Status:

Not required. WinCE BSP does not use the TOG_DIS bit.

ENGcm10189 eCSPI Burst completion by SSB signal in Slave mode is not functional

Description:

According to the eCSPI specifications, when eCSPI is set to operate in the Slave mode (CHANNEL_MODE[x] = 0), the SSB_CTRL[x] bit controls the behavior of burst completion.

In the Slave mode, the SSB_CTRL bit controls the behavior of SPI burst completion as follows:

- 0—SPI burst completed when (BURST_LENGTH + 1) bits are received
- 1—SPI burst completed when SSB input negated

Also, in BURST_LENGTH definition, it is stated “In the Slave mode, this field takes effect in SPI transfer only when SSCTL is cleared.”

However, the mode SSB_CTRL[x] = 1 is not functional in Slave mode. Currently, BURST_LENGTH always defines the burst length.

According to the SPI protocol, negation of SSB always causes completion of the burst. However, due to the above issue, the data is not sampled correctly in RxFIFO when {BURST_LENGTH+1}mod32 is not equal to {actual burst length}mod32.

Therefore, setting the BURST_LENGTH parameter to a value greater than the actual burst does not resolve the issue.

Projected Impact:

Slave mode with unspecified burst length cannot be supported due to this issue. The burst length should always be specified with the BURST_LENGTH parameter and the SSB_CTRL[x] should be set to zero.

Workarounds:

There is no workaround except for not using the SSB_CTRL[x] = 1 option in the Slave mode. The accurate burst length should always be specified using the BURST_LENGTH parameter.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not implemented because eCSPI slave mode is not supported in the Linux BSP driver.

WinCE BSP Status:

Not required. WinCE BSP does not support slave mode.

ENGcm11038 EMI2.5: Read from M4IF Watermark status registers may have wrong result

Description:

Due to non proper clock synchronization implementation, the WMIS0 and the WMIS1 watermark status registers can return wrong value.

The probability is low.

Projected Impact:

The watermark status and the interrupt indications can be read wrongly. The status bits can be read as zero instead of one (Reading one as zero does not likely to occur).

Workarounds:

In the real applications, the Read command is to be issued after a watermark interrupt is issued to the Cortex. After the watermark interrupt is generated, the appropriate status register indicators should read as one to indicate the violated watermark region.

A workaround to the issue is to check the data that is read, and repeat the read command in case all the register bits values are read as zero.

Proposed Solution:

No fix scheduled

ENGcm11245 EMI: WEIM 8-bit memory devices support clarification

Description:

8-bit memory devices are supported by the WEIM interface, which is connected only to the EIM_D[31:24] pads.

Connection to the EIM_D[23:16] pads is not supported.

This erratum clarifies the statement in the *i.MX53 Applications Processor Reference Manual* (MCIMX53RM), that only 16-bit and 32-bit memory devices are supported by the WEIM interface.

Projected Impact:

WEIM 8-bit memory devices are supported according to above description.

Workarounds:

None

Proposed Solution:

No fix scheduled. A clarification is added in the reference manual.

Linux BSP Status:

No software workaround

WinCE BSP Status:

No software workaround

ENGcm11786 EMI: EIM BCLK in DEBUG mode is not functional when GBCD is zero and BCD is not zero

Description:

EIM has a debug mode that enables BCLK to be a free running clock. This mode is selected by writing 1 to the BCM bit of the EIM's WCR register.

- When BCM is 1, this clock can be divided by configuring the GBCD field.
- When BCM is 0, this clock can be divided by configuring the BCD field.

Due to this error, when BCM is set, GBCD is not 0 and BCD = 0, the result will be a non-active clock.

Projected Impact:

Due to this erratum, the software specific combination of BCM,GBCD, & BCD is not working.

Workarounds:

This can be bypassed by configuring the BCD of all the active chip selects (CS) to be equal to GBCD.

Proposed Solution:

No fix scheduled

ENGcm11028 EPIT: Possibility of additional pulse on src_clk when switching between clock sources

Description:

There is a possibility of an extra pulse on SCLK in the EPIT, when switching between the clock sources.

Projected Impact:

It can result in an incorrect counter increment in the EPIT.

Workarounds:

Clock source should be changed only when the EPIT is disabled. A way to accomplish the same is as follows:

1. Disable EPIT—EPITCR[0] = 0 (EN = 0), that is, disable EPIT
2. Disable EPIT output—EPITCR[23:22] = 00 (OM = 00)
3. Disable EPIT capture interrupt—EPITCR[2] = 0 (OCIEN = 0)
4. Change clock source—EPITCR[25:24] (CLKSRC), determines which clock source is selected for running the counter
5. Clear status register—EPITSR[0] (OCIF), this is a write one to clear register
6. Configure EPIT to start count once enabled from load value—EPITCR[1] = 1 (ENMOD = 1)
7. Re-enable EPIT EPITCR[0] = 1 (EN = 1), that is, enable EPIT
8. Reconfigure output and interrupt

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not implemented because EPIT is not used now.

WinCE BSP Status:

Not required. BSP does not switch EPIT clk src.

ENGcm11052 eSDCTL: Auto power down issues

Description:

Two issues are observed with respect to the automatic power down in eSDCTL:

1. PWDT cycle amount in the specification should be divided by 2.
2. The PWDT should be individually enabled for each chip select, to gain maximum power saving. However a bug is observed, which resulted in read operation from one chip select, preventing the other chip select from entering the power down mode. Only if both chip selects are idle, the power down mode is observed.

Projected Impact:

The issue impacts the power saving optimization.

Workarounds:

None

Proposed Solution:

No fix scheduled

ENGcm11213 eSDHC: Glitch is generated on card clock with software reset or clock divider change

Description:

A glitch may occur on the SDHC card clock when the software sets the RSTA bit (software reset) in the system control register. It can also be generated by setting the clock divider value. The glitch produced can cause the external card to switch to an unknown state. The occurrence is not deterministic.

Projected Impact:

Potential disruption of SD card operation.

Workarounds:

A simple workaround is to disable the SD card clock before the software reset, and enable it when the module resumes the normal operation. The Host and the SD card are in a master-slave relationship. The Host provides clock and control transfer across the interface. Therefore, any existing operation is discarded when the Host controller is reset.

The recommended flow is as follows:

1. Software disable bit[3] of the System Control register
2. Trigger software reset and/or set clock divider
3. Check bit[3] of the Present State Register for stable clock
4. Enable bit[3] of the System Control register.

Using the above method, the eSDHC cannot send command or transfer data when there is a glitch in the clock line, and the glitch does not cause any issue.

Proposed Solution:

The *i.MX53 Applications Processor Reference Manual* (MCIMX53RM) is updated with the workaround procedure, that requires the SDCLKEN bit to be disabled before setting software reset and/or setting clock divider, and then enabling the SDCLKEN bit, after SDSTB bit is set high (Clock stable flag).

Linux BSP Status:

Not required because Linux driver avoids this issue.

WinCE BSP Status:

Partially implemented. Full implementation in next release.

ENGcm11116 eSDHCv2/eSDHCv3: ADMA transfer error when the block size is not a multiple of four

Description:

Issue in eSDHC ADMA mode operation. The eSDHC read transfer is not completed when block size is not a multiple of 4 in transfer mode ADMA1 or ADMA2. The eSDHC DMA controller is stuck waiting for the TC bit in the interrupt status register.

The following examples trigger this issue:

1. Working with an SD card while setting ADMA1 mode in the eSDHC
2. Performing partial block read
3. Writing one block of length 0x200
4. Reading two blocks of length 0x22 each. Reading from the address where the write operation is performed. Start address is 0x512 aligned. Watermark is set as one word during read. This read is performed using only one ADMA1 descriptor in which the total size of the transfer is programmed as 0x44 (2 blocks of 0x22).

Projected Impact:

The issue exists only when the block size is not a multiple of 4 in transfer mode ADMA1 or ADMA2.

Workarounds:

When the ADMA1 or ADMA2 mode is used and the block size is not a multiple of 4, the software should set the block size to:

$$4 \cdot \left\lceil \frac{\text{block size}}{4} \right\rceil$$

In other words, the block size should be rounded to the next multiple of 4 bytes. In case of write, the software should add the corresponding number of bytes at each block end, before the write is initialized. In case of read, the software should remove the dummy bytes after the read is completed.

For example, if the original block length is 22 bytes, and there are several blocks to transfer, the software should set the block size to 24. The following data is written/stored in the external memory:

```
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
2 Bytes valid data + 2 Byte dummy data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
4 Bytes valid data
2 Bytes valid data + 2 Byte dummy data
```

In this example, 48 (24 x 2) bytes are transferred instead of 44 bytes. The software should remove the dummy data.

Alternatively, the PIO mode can be used if the block size is non-4 byte aligned.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because ADMA mode is not enabled.

WinCE BSP Status:

Not required. BSP does not have such case.

ENGcm11115 eSDHCv2/eSDHCv3: Problem when ADMA2 last descriptor is LINK or NOP

Description:

ADMA2 mode in the eSDHC is used for transfers to/from the SD card. There are three types of ADMA2 descriptors: TRANS, LINK or NOP. The eSDHC has a problem when the last descriptor (which has the End bit '1') is a LINK descriptor or a NOP descriptor.

In this case, the eSDHC completes the transfers associated with this descriptor set, whereas it does not even start the transfers associated with the new data command. For example, if a WRITE transfer operation is performed on the card using ADMA2, and the last descriptor of the WRITE descriptor set is a LINK descriptor, then the WRITE is successfully finished. Now, if a READ transfer is programmed from the SD card using ADMA2, then this transfer does not go through.

Projected Impact:

Limitation for configuring the last descriptor as LINK or NOP.

Workarounds:

Software workaround is to always program TRANS descriptor as the last descriptor.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because ADMA mode is not enabled.

WinCE BSP Status:

Not required. BSP does not have such case.

ENGcm11186 eSDHCv2/eSDHCv3: eSDHC misses SDIO interrupt when CINT is disabled

Description:

An issue is identified when interfacing the SDIO card. There is a case where an SDIO interrupt from the card is not recognized by the hardware, resulting in a hang.

If the SDIO card lowers the DAT1 line (which indicates an interrupt) when the SDIO interrupt is disabled in the eSDHC registers (that is, CINTEN bits in IRQSTATEN and IRQSIGEN are set to zero), then, after the SDIO interrupt is enabled (by setting the CINTEN bits in IRQSTATEN and IRQSIGEN registers), the eSDHC does not sense that the DAT1 line is low. Therefore, it fails to set the CINT interrupt in IRQSTAT even if DAT1 is low.

Generally, CINTEN bit is disabled in interrupt service.

The SDIO interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.
2. Reset the interrupt factors in the SDIO card and write 1 to clear the CINT interrupt in IRQSTAT.
3. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

If a new SDIO interrupt from the card occurs between step 2 and step 3, the eSDHC skips it.

Projected Impact:

The issue is relevant only for the SDIO card interrupt usage.

Workarounds:

The workaround interrupt service steps are as follows:

1. Clear CINTEN bit in IRQSTATEN and IRQSIGEN.
2. Reset the interrupt factors in the SDIO card and write 1 to clear CINT interrupt in IRQSTAT.
3. Clear and then set D3CD bit in the PROCTL register. Clearing D3CD bit sets the reverse signal of DAT1 to low, even if DAT1 is low. After D3CD bit is re-enabled, the eSDHC can catch the posedge of the reversed DAT1 signal, if the DAT1 line is still low.
4. Re-enable CINTEN bit in IRQSTATEN and IRQSIGEN.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Not implemented yet

ENGcm11406 eSDHCv3: DLL_STS_REF_LOCK status bit does not indicate when DLL is locked

Description:

DLL_STS_REF_LOCK bit in DLL Status Register (DLLSTS) does not indicate the reference DLL lock status properly. The DLL_STS_REF_LOCK bit function indicates that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays.

Because the timing delay of one tap cell in DLL is too short, a two cell delay period is not wide enough to allow phase_n and phase_p of the reference clock to straddle the high-pulse at the same time. As a result, the DLL_STS_REF_LOCK status bit cannot indicate the actual status.

Projected Impact:

The DLL functionality itself is not impacted, only the DLL_STS_REF_LOC indicator is not working properly.

Workarounds:

Do not use the DLL_STS_REF_LOC as indicator. The DLL_STS_SLV_LOCK status bit is sufficient indicator for the software. The DLL_STS_SLV_LOCK represents slave delay-line lock status. It indicates that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.

Proposed Solution:

No fix scheduled

ENGcm11214 FEC: Fast Ethernet Controller (FEC) accesses to NAND Flash Controller (NFC) does not work

Description:

The FEC only creates wrap burst accesses upon data access (non-buffer descriptor). The NFC under EMIv2 cannot handle such accesses from the MAX (M2) input port.

Projected Impact:

The FEC cannot use the NFC to access memory.

Workarounds:

The FEC should use the DDR for its buffer.

Proposed Solution:

No fix scheduled

Linux BSP Status:

The case does not occur in Linux BSP.

WinCE BSP Status:

Implemented.

ENGcm11029 GPT: Possibility of additional pulse on src_clk when switching between clock sources

Description:

There is a possibility of an extra pulse on SCLK in the GPT when switching between the clock sources.

Projected Impact:

The bug can produce an incorrect counter increment in the GPT when switching between the clock sources.

Workarounds:

Changing the clock source should only be done when the GPT is disabled. A way to accomplished this is as follows:

1. Disable GPT—Write 1'b0 to EN bit of GPTCR
2. Disable interrupts—Write 6'b000000 in Bits [5:0] of GPTIR
3. Configure Output Mode to unconnected/ disconnected—Write zeros in OM3, OM2, OM1 in GPTCR
4. Disable Input Capture Modes—Write zeros in IM1,IM2 in GPTCR
5. Change clock source CLKSRC in GPTCR
6. Clear Status register—Write 003F in GPTSR
7. Set ENMOD in GPTCR
8. ENABLE GPT—Write 1'b1 to EN bit of GPTCR. The GPTSR should not be read immediately after changing the clock source (a wait of at least one SCLK is required).

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because GPT parent is not changed in current code.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER7.

ERR007080 LDO: On-chip LDO regulators may not enable or have a delayed output on power up

Description:

In certain applications, the on-chip LDOs may not power up correctly. When the issue occurs, both internal LDO regulators do not drive the VDD_ANA_PLL and VDD_DIG_PLL supply outputs. The potential for this outcome is aggravated at high temperatures (>50-60°C) and slower VCC supply ramp times.

Impacted units do not function properly due to incorrect operation of the TEST_MODE input. The device's offset latch circuit is designed to output a logic level zero, on application of VCC (logic power) and in the absence of NVCC_RESET (I/O power). The TEST_MODE circuit outputs an erroneous logic level one on application of VCC in the impacted units. This has the unintended effect of putting the device into test mode state which powers off the LDO regulators.

On impacted units, if the applied NVCC_RESET is not sourced from the internal LDO regulator then the output of the LDO is delayed till NVCC_RESET is applied. This LDO delay corresponds to the time it takes to bring up NVCC_RESET after VDD_REG.

Projected Impact:

The impact depends on the system's power sequencing and supply connections as follows:

- If the on-chip LDO regulator powers NVCC_RESET, it prevents the correct TEST_MODE input value from reaching the LDO enable logic, thereby creating a deadlock situation in the impacted units. Both LDO regulators do not enable and drive the VDD_ANA_PLL and VDD_DIG_PLL supplies.
- If an external regulator powers NVCC_RESET, and the power-up sequence complies with the data sheet then the issue does not occur. Externally supplying and correctly sequencing the NVCC_RESET supply allows the TEST_MODE logic to achieve the correct state. Thus, the LDOs are enabled.
- If an external regulator powers NVCC_RESET, and comes up after VDD_REG , then a delayed LDO output enable is observed. Once NVCC_RESET comes up, it correctly resolves the state of the TEST_MODE input, thus enabling the internal LDO regulators. This delay is not a system issue since LDO's are fully functional and POR_B remains low while other supplies are sequenced.

Workaround:

Users must ensure that the NVCC_RESET supply is powered externally (not from the i.MX53 LDO VDD_ANA_PLL supply output) after VCC is stable and before other I/O supplies (NVCC_xxx) are powered up.

Silicon Fix:

A Silicon revision is planned. Please contact your Freescale representative for details.

With revised silicon, if NVCC_RESET power is removed or interrupted, a POR is generated.

.

ENGcm11127 M4IF: Step-by-step mechanism violates AXI protocol

Description:

If the step-by-step mechanism is enabled while the M4IF is operational (there are transactions in the internal buffers), there can be a situation where one or more of the arbitrations' AXI protocol is violated in the “write address channel” or “read address channel.”

Projected Impact:

The M4IF violates the AXI protocol and crashes if step-by-step is set ON or OFF during the run-time.

Workarounds:

Before entering the step-by-step mode, configure the EMI to the software LPMD and then read the LPACK register. The LPACK register indicates that the M4IF is idle and the step-by-step can be enabled. The difference between the regular procedure of LPMD and this procedure is that the EMI clocks remain ON.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required

WinCE BSP Status:

Not required

ENGcm10682 ENGcm10940 M4IF power-saving mode should not be enabled before DDR is configured

Description:

Enabling the power-saving mode in M4IF before configuring the DDR module causes deadlock during DDR configuration.

Projected Impact:

None

Workarounds:

Initialization must be performed first, followed by enabling the M4IF power saving.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP.

ENGcm11203 M4IF: Reading M4IF status registers of an inactive AXI master or slave stalls entire system

Description:

Reading the M4IF status registers of an inactive AXI master or slave ports stalls the entire IC system. This occurs when a specific master or slave clock is not provided to the M4IF, but the ARM/JTAG tries to read that master's status bits.

When the master or slave clock is not active, the read request fails to propagate the status bits that go through synchronization (IPG_CLK to *_clk), acknowledge never comes back, and the entire chip's IP bus is stuck.

Some M4IF status registers bundle the status of several masters or slaves. If one of the masters or slaves is inactive, accessing such a register stalls the entire system, even if the user is interested only in the status of other active ports.

Table 4 lists the impacted registers and the clock signals that should be active for read access to succeed.

Table 4. List of Impacted Registers

Registers	Required Master/Slave to be Active for Read Action to Succeed
MDCR	fast, slow, int1, int2
WMIS0	fast
WMIS1	fast
MLEN	m0, m1, m2, m3, m4, m5, m6, m7, fast, slow, int1, int2
FDPS	fast
SSRL0	fast
SSRL1	fast
SSRH0	fast
SSRH1	fast
MDSR0	The arbitration domain selected by MDCR/RARB
MDSR1	The arbitration domain selected by MDCR/RARB
MDSR2	The arbitration domain selected by MDCR/RARB
MDSR3	The arbitration domain selected by MDCR/RARB
MDSR4	The arbitration domain selected by MDCR/RARB
MDSR5	The arbitration domain selected by MDCR/RARB
MDSR6	The arbitration domain selected by MDCR/RARB
MDSR7	The arbitration domain selected by MDCR/RARB
MDSR8	The arbitration domain selected by MDCR/RARB
SBS0	The arbitration domain selected by MDCR/RARB
SBS1	The arbitration domain selected by MDCR/RARB

Table 4. List of Impacted Registers (continued)

Registers	Required Master/Slave to be Active for Read Action to Succeed
PSM0	m0,m1
PSM1	m2,m3
PSM2	m4,m5
PSM3	m6,m7
MDCR	fast, slow, int1, int2
MCR0	fast, slow, int1, int2

Projected Impact:

No impact on regular functionality. The status registers are usually accessed for debugging purposes.

Workarounds:

Enable the relevant M4IF masters or slaves clocks for the status read.

Proposed Solution:

No fix scheduled

ENGcm11215 NFC: 8-Sym ECC mode does not work with 512 byte page x16 bus NAND Flash

Description:

When ECC_MODE = 1 the 8-sym error detection and correction does not work with 512 byte main area ×16 NAND Flash.

Projected Impact:

The 8-sym ECC mode does not work with ×16 bus width NAND. This mode works with ×8 bus width NAND. This restricts 8-sym ECC operation to 8-bit NAND devices.

Workarounds:

None

Proposed Solution:

No fix scheduled. The reference manual is updated accordingly.

Linux BSP Status:

No software workaround required.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm10288 NFC: Copy back function destination address restriction

Description:

The copy back feature of the NFC module does not work as expected. When trying to copy a page from source address to destination address, the destination address is always the successive page of the source address.

Projected Impact:

The copy back cannot be done to an address that is not successive to the source address.

Workarounds:

There are two options:

- Perform the copy back through Atomic operations
- Instead of copy back, perform a read operation followed by a write operation

Both options affect the overall performance.

Proposed Solution:

No fix scheduled. Clarification added to reference manual.

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm11124 NFC: Block write-protect does not support lock-tight

Description:

NFC offers a block-write-protect feature in which only a range of pre-defined blocks can be modified. This range of blocks can be set to UNLOCK (blocks that can be modified), LOCK (blocks that cannot be modified) or LOCK_TIGHT (blocks that cannot be modified and the range cannot be changed).

Though switching to LOCK_TIGHT mode, the range of blocks can still be modified.

Projected Impact:

Lock cannot be trusted to prevent data from being overwritten. As the software does not use this feature (both WinCE and LINUX), there is no impact.

Workarounds:

As there is no difference between LOCK and LOCK_TIGHT modes, the software should not use LOCK_TIGHT mode.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm11217 NFC: Block write-protect does not work in automatic operations

Description:

NFC offers a block-write-protect mechanism. Only a configurable range of NAND blocks can be modified. Any erase/program operations on the blocks outside this range should be blocked by NFC. This mechanism does not work in automatic program and in automatic erase.

Projected Impact:

As there is no workaround for this bug, the write protect mechanism cannot be used.

Workarounds:

It should be handled in the software, as it is done now for the Windows and Linux drivers.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm11182 NFC: Copy-back does not work properly when addr_op = 01

Description:

When working with addr_op = 01, and trying to perform a copy-back operation, the NFC ignores the destination address configured, and copies the page to “source address” +1. Additionally, the following automatic operation is carried out from address_register1 instead of address_register0.

Projected Impact:

Automatic operation is carried out from address_register1 instead of address_register0.

Workaround:

If the system requires to work in addr_op = 01, switch to addr_op = 11, before the copy-back operation, and switch back after the copy-back operation is complete.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

No plan to support as there is no NAND Flash on EVK.

ENGcm11126 NFC: Software reset does not work properly under certain conditions

Description:

The software reset (setting the NFC_RST bit in NFC and then sending reset command 0xFF to NAND Flash) does not work correctly under the following conditions:

- Reset between the read operations
- With atomic program operation, the RESET command is not being issued
- Auto program operation—If reset occurs after writing data to the NFC and before the write confirm command, the CACK bit is not set after setting CREQ

Projected Impact:

The software reset does not work consistently, but the failure conditions are not common in real world applications.

Workarounds:

Do not to apply software reset for the above conditions.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not have such case.

ENGcm11218 NFC: Status read does not occur at the end of the program, with RBB_MODE = 1

Description:

After automatic Program operation the NFC is expected to perform a status read and store the status in STATUS_SUM register. With RBB_MODE = 1 (that is, NFC waiting using BSY_B signal), the status read does not occur at the end of the Program/CopyBack0 and CopyBack1 operations.

Projected Impact:

The status is not correct.

Workarounds:

Workaround for this erratum:

- In the Automatic mode, use RBB_MODE = 0
- If RBB_MODE = 1 is to be used, then after the Automatic program, Automatic CopyBack0 and Automatic CopyBack1 operations, Status read should be done explicitly.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

WinCE BSP Status:

Workaround implemented in WinCE BSP release ER9.

ENGcm11219 NFC: Misses read data when working in Symmetric mode with clock ratio 1:2, and using a 16-bit Flash bus width

Description:

When working in symmetric mode with a clock ratio 1:2, and using a 16-bit Flash bus width, the NFC reads data and organizes it using a shift in the internal RAM in such a way that the last 16 bits of the data block being transferred are not written to the memory.

Projected Impact:

Wrong data may be read.

Workarounds:

Avoid combining the following parameters/conditions:

- 16-bit flash bus width
- Symmetric mode
- 1:2 clock-ratio

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because BSP does not have such case.

WinCE BSP Status:

Not required because the 16-bit NAND is not supported with current WinCE BSP.

ENGcm11220 NFC: Cannot reach entire address space when addr_op = 1 or 3

Description:

When the NFC is configured to addr_op = 1 or 3

- If num_of_devices = 0–1, then the LSB of the page/blocks address section of the addr_group bits is not used for address generation.
- If num_of_devices = 2–3, then the 2 LSB's of the page/blocks address section of the addr_group bits is not used for address generation.
- If num_of_devices = 4–7, then the 3 LSB's of the page/blocks address section of the addr_group bits is not used for address generation.

Projected Impact:

Working in these addr_op modes has some limitations on the size of the devices (depending on the number of devices). Such large devices do not exist at this time. This can be an issue when using larger devices that may become available in future.

NOTE

addr_op = 0 works as designed with any combination.

Workarounds:

Do not use the combination of parameters/modes described above.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because such devices are not available yet.

WinCE BSP Status:

Not required because such devices are not available yet.

ENGcm11221 NFC: ECC mechanism may fail to report uncorrectable error situation

Description:

The NFC error correction mechanism is based on BCH error correction codes. The BCH error correction code allows correction up to a particular number of errors (T = 4 or 8 depending on the configuration), and detection in case of higher number of error bits. In a situation where the number of errors (NOBER) is larger than the number of errors the NFC can correct (T), the NFC should report on uncorrectable error.

Due to a theoretical limitation, the BCH code can fail in detecting the error when the number of actual error bits is much larger than the number of correctable bits. The failure depends on the location of the errors (related to Hamming distance), and its probability is very small. The detection failure probability can be calculated using the formula as follows:

$$P_e = \frac{1}{T! \cdot 2^T}$$

where T is number of correctable errors.

- For 4-bit ECC, the calculated probability for a decoder error is 1 : 384
- For 8-bit ECC, the calculated probability for a decoder error is 1 : 10,321,920

In case of failure, the NFC reports only on T or less errors (NOBER=T) and tries to correct them. As a result the data is damaged and the NFC does not report on it.

Projected Impact:

There is a small probability for failure in detection of the errors by the ECC mechanism due to theoretical limitation of the ECC code.

Workarounds:

To reduce the failure significantly, treat the case where NFC reports NOBER = T as uncorrectable error. This implies that if the number of errors is equal to T, the software must invalidate the block.

It is a common practice for the software to mark the block as bad in advance when the number of reported errors approaches T.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No complete software workaround.

WinCE BSP Status:

Software workaround implemented. It can alleviate the issue, but cannot completely solve it. Such use case (4-bit ECC) is rare under current market environment.

ENGcm11002 NFC can miss the sampling of the ready/busy signal ($\overline{R/B}$) when RBB_MODE = 1

Description:

The NFC may not properly sample the ready/busy ($\overline{R/B}$) signal under the following conditions:

1. RBB_MODE = 1 (Ready-Busy mode 1 - NFC monitors ready-busy status by checking NANDF_RBx ($\overline{R/B}$) signals)
2. ADD_OP = 01 (Addressing Option 01 - NFC uses only address_group0 with single NAND device
3. enfc_clk period is less than $t_{wb}/3$ (t_{wb} is the period from $\overline{WE}$ write enable signal HIGH to $\overline{R/B}$ ready/busy signal assertion)

If RBB_MODE = 1 and ADD_OP = 01, the NFC may miss the sampling of the ready/busy signal ($\overline{R/B}$) if the enfc_clk is too fast. According to the NAND flash protocol, the NAND device should enter the busy mode and assert the $\overline{R/B}$ signal (driving $\overline{R/B}$ signal to zero) after the maximum time period of t_{wb} after the deassertion of the $\overline{WE}$ signal (driving write enable to one).

The NFC samples the $\overline{R/B}$ signal after a fixed time of 3 enfc_clk cycles after the deassertion of $\overline{WE}$. Typically, the t_{wb} period is 100 ns. When the enfc_clk is set to less than 33 ns period, the NFC may miss sampling the $\overline{R/B}$.

Projected Impact:

Violation of NAND interface protocol can result in data corruption. The implementation of the proposed workaround may have insignificant performance impact.

Workarounds:

The recommended workaround is to set RBB_MODE = 0. This also frees up the NANDF_RBx pads for other usages. The NFC monitors the ready-busy status by performing a status-read command.

Another option is to work with ADD_OP other than 01, allowing to work automatically with a single device other than device0.

See, NAND_FLASH_CONFIG Register Field Descriptions table in the *i.MX53 Applications Processor Reference Manual* (MCIMX53RM) for detailed modes description.

If the above two options are not feasible, the enfc_clk frequency can be reduced such that clock period is larger than $t_{wb}/3$. This results in small performance degradation as the supported ONFI1.0 NAND can run at up to 40 MHz.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround implemented in Linux BSP release 09.12.00.

ENGcm11053 SDMA multi-page read from the NFC, when the WEIM is operating, can result in data corruption or EMI hanging

Description:

Data corruption or EMI hanging can occur on SDMA Multi-Page read from the NFC in case of simultaneous read access by other master from WEIM. Use of SDMA is relevant for a case of multiple pages read from the NFC (that is, setting NUM_OF_ITERATION to a value greater than zero and setting NO_SDMA to zero in NFC for automatic interleave mode).

Projected Impact:

SDMA read access from NFC may be corrupted in case other masters perform read from WEIM at same time.

There are no issues in following cases:

- ARM accesses the NFC when WEIM is accessed by any other master in the system
- SDMA accesses the NFC when the WEIM is idle
- NFC is idle and WEIM is accessed by any master in the system

Workarounds:

Avoid the above described situation. Note that, in current WinCE and Linux BSP releases from Freescale, the SDMA Multi-Page mode read from NFC is not activated.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because the SDMA Multi-Page mode read from NFC is not activated.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm11180 Grounding nonfunctional UHVIO IO pads power rails can cause malfunction in other UHVIO IO cells

Description:

If the power supply of an unused UHVIO pad power group is pulled down, it can cause malfunction in the other functional UHVIO IO cells. The UHVIO IO cells (that support signal levels above 3 V) are used in interfaces such as SD card, NAND Flash. For example, grounding SD2 power supply rail through a small resistor (say 470 Ω) causes boot issues from the SD1 interface. This is caused due to a particular HVIO IO cell design limitation.

Projected Impact:

Pay attention to this limitation during board design. An unintentional grounding can occur because of nonfunctional or disabled voltage supply device connected to the UHVIO based interface VCC line.

Workarounds:

Take measures to avoid such cases. Leaving the unused interface supply open, does not cause any issue. The issue occurs only when a supply is grounded through a small resistor. The best design practice is to apply power to all the supply rails.

Proposed Solution:

No fix scheduled

ENGcm11642 UHVIO pads automatic supply voltage level detect function may not work for voltage between 1.95 V to 3.1 V

Description:

UHVIO pads have an automatic power supply voltage level detector mechanism that should detect if the interface voltage is high (3.0 V – 3.6 V) or low (1.65 V – 3.1 V), and configure the pad accordingly.

This detector may not work correctly for voltage levels between 1.95 V and 3.1 V. It detects this range as high voltage when it should be low. This impacts the pad timing performance, adding a propagation delay of up to 2.5 ns.

Projected Impact:

UHVIO pads are used by several interfaces such as NAND Flash, SDHC, EIM, FEC, Keypad, PATA, CSI, and some GPIOs. For general functionality the impact is not significant because the pads can be configured appropriately by setting the IOMUX pad control register bit VDOEN[1] to manual voltage selection and the HVEOVERWRITE bit to 1. However, during the boot, the pads are configured by default to enable the automatic supply voltage detection. As a result, UHVIO pad base interfaces cannot use the fast boot mode if they are in the 1.95 V – 3.1 V voltage range. This limitation applies particularly to NANDF and SDHC interfaces.

Workarounds:

In case the UHVIO based interface operates at voltage level in the 1.95 V to 3.1 V range, configure the pads for manual voltage level selection by setting the IOMUX pad control register bit VDOEN[1] for manual voltage selection and the HVEOVERWRITE bit to 1.

To guarantee no timing issues occur during boot on an external interface operating at a voltage level in the 1.95 V to 3.1 V range, disable the fast boot mode by burning the corresponding e-fuse. E-fuses exist in both NAND Flash and SDHC boot modes, and select either a fast or slow boot option.

Proposed Solution:

No fix scheduled

1. Note the erratum ENGcm11851 related to VDOEN bit polarity. The polarity of the bit in i.MX53 revision 2.0 is different from revision 1.0.

ENGcm11851 ROM (Boot): Boot from SATA fails when internal clock mode is used

Description:

SATA can work in two clock modes: internal and external. Due to an error in the ROM code, boot from SATA when using the internal clock mode is not working. SATA can be used in internal clock mode if boot is done from another source.

Projected Impact:

Boot from SATA when working in internal clock mode is not working.

Workarounds:

1. Boot from SATA when working in external clock mode can be used.
2. Use Low Power Boot Mode (LPB) and force low power condition using pull-ups or pull-downs. LPB is used in the following way:
   - Blow BT_LPB fuse.
   - Do one of the following options:
     - Pull-up PATA_DIOW pad to imitate low power condition and blow BT_LPB_POLARITY fuse
     - Pull-down PATA_DIOW pad to imitate low power condition and avoid blowing BT_LPB_POLARITY

Proposed Solution:

No fix planned

ENGcm10971 RTICv3 memory region unlock feature can cause the RTICv3 to hang

Description:

The RTICv3 feature that allows the TrustZone software to disable the run-time check of the selected memory regions (region unlock), can cause the RTICv3 to hang and stop the run time integrity check of other regions.

Projected Impact:

The RTICv3 feature that allows the TrustZone software to disable run-time check of the selected memory regions cannot be utilized.

Workarounds:

None. Avoid using the memory region unlock feature in the TrustZone code.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required. BSP does not use this feature.

WinCE BSP Status:

Not required. BSP does not use this feature.

ENGcm10363 SAHARA/CCM: Frequency ratio restriction for AHB and IP buses in SAHARA

Description:

SAHARA does not work properly if AHB:IP buses clock ratio is 1:1. It works fine with AHB:IP buses clock ratio 2:1.

Projected Impact:

Adds limitation on system configuration for the clock ratio of 1:1 between AHB and IP buses.

Workarounds:

Avoid using a ratio of 1:1 between AHB and IP buses clock frequencies. Other software workarounds are considered, but not yet confirmed.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required. Be aware of the limitation. Avoid using a ratio of 1:1 between the AHB and IP buses clock frequencies.

WinCE BSP Status:

SAHARA is not used in the current BSP release.

ENGcm10417 SATA: IS.INFS bit is not set when ERR_C is set

Description:

The AHCI specification states that the IS.INFS bit should be set when ERR_C is set, which does not occur. ERR_C is set when a Phy_not_ready condition is detected during non-data FIS reception or transmission.

Projected Impact:

None. In addition to ERR_C, a Phy_not_ready condition also causes the IS.PRCS bit to be set. IS.PRCS = 0 and SSTS.DET = 0 (indicates device disconnect) supersede other errors and should be handled by the software. Setting IFS/INFS is redundant and does not give any more useful information to the software.

The severity is low. Medium probability.

Workarounds:

None

Proposed Solution:

No fix scheduled

ENGcm10975 SATA: Unknown FIS with incorrect PMP field received

Description:

When an unknown FIS (UFIS) is received and the PMP field is incorrect (does not correspond to the command header CH.PMP field), both PDMA and TSM state machines lock-up, and IS.IPMS interrupt is generated.

Projected Impact:

A UFIS is received in the RxFIFO and the IS.IPMS bit is set, generating IPMS intrq. Therefore, PDMA and TSM state machines lock-up.

The severity is medium. The probability of occurrence is low. Receiving an unknown FIS is an error scenario. This should not happen in the normal operation.

Workarounds:

Host software issues port reset (COMRESET) or global reset, if IS.IPMS interrupt is received.

Proposed Solution:

No fix scheduled

ENGcm10983 SATA: Erroneous detection of Read Overflow condition

Description:

Under a certain condition (small data transfer of 20 bytes with two Data FISes (12 bytes + 8 bytes + “end-status” is delayed a little) and four PRDs (4 + 4 + 4 + 8 bytes)), it is possible that a read overflow condition can be detected erroneously when such a condition is not present. As a result the P#IS.OFS is set.

Projected Impact:

A P#IS.OFS interrupt is generated erroneously. The Host controller is not affected otherwise. So the normal operation can continue.

The severity is low. The probability is low.

Workarounds:

Software can either retry the command or generate COMRESET.

Proposed Solution:

No fix scheduled

ENGcm10982 SATA: Soft Reset not sent when issued by the software during on-going transfer

Description:

The SRST FIS is not read from the system memory, when the ST bit clears during an on-going transfer and the core locks up. This happens only during certain conditions (for example, when a DMA read transfer takes a long time due to a large burst being converted into multiple single transfers).

There are two parts to this problem:

1. TSM locks up in HT_PIOOTRANS2 (PIO) or in HT_DMAOTRANS2 (DMA) when software wants to send a SRST and clears the CMD.ST bit
2. PDMA unsuccessfully requests an SRST command header when the current read transfer is not finished and p_dma_req = 1

Projected Impact:

The PDMA/TSM module lockup.

The severity is medium. The probability is low.

Workarounds:

The software should use Port Reset (COMRESET) instead.

Proposed Solution:

No fix scheduled

ENGcm11018 SATA: Problems with phy_reset when Staggered Spin-Up supported

Description:

Under certain conditions (combination of core clock values and software delays), the core can generate a phy_reset indefinitely.

It occurs during one of the following cases:

1. If Global reset causes CMD.SUD = 0, then the software writes CMD.SUD = 1 to spin-up to the device
2. Under normal operation, when CMD.SUD = 1 and when the software issues a port reset by toggling SCTL.DET = 0 $\rightarrow$ 1 $\rightarrow$ 0

Projected Impact:

The top-level port phy_reset is asserted indefinitely. This is a synchronous reset, typically used only when the PHY handles TX OOB.

The severity is low. The probability is low.

Workarounds:

Port reset/COMRESET after time-out.

Proposed Solution:

No fix scheduled

ENGcm11084-1 SATA: PMP field checked when CMD.PMA = 0

Description:

The SATA AHCI specification states that the software should issue a soft reset with 0xF in the PMP field, to check if a port multiplier is connected to the Host controller. If a device is connected directly to the Host controller, without a port multiplier, the returned FIS has a value of 0x0 in the PMP field. As the values of the PMP field in the Soft Reset FIS and the returned Signature FIS from the device are different, the Host controller reports an error and sets the IS.IPMS bit.

The PDMA module is supposed to check the PMP field of the incoming FIS only when CMD.PMA = 1. As CMD.PMA is set to zero initially, the controller should accept the signature FIS from the device even with PMP = 0xF. Currently, PMA is not used/ignored.

Projected Impact:

If a port multiplier is connected, problem does not arise. When the software attempts to enumerate the port multiplier, the port multiplier returns a FIS with 0xF in the PMP field, which is expected by the controller. If a device is connected directly to the Host without a port multiplier, the PMP field in the returned FIS contains 0x0, as opposed to 0xF in the soft reset FIS issued by the software. This causes the IS.IPMS bit to be set, and the TFD.STS.BSY bit is never cleared.

The severity is medium. The probability is low.

Workarounds:

When IS.IPMS is set and TFD.STS.BSY is not cleared, no further transfers can be made until a COMRESET is issued. Determine the presence/absence of the Port Multiplier using IPMS intrq. If it is set, the PM is not attached and SRST with PMP = 0h must be issued.

Proposed Solution:

No fix scheduled

ENGcm11084-2 SATA: Data FIS received when CMD.ST = 0

Description:

If CMD.ST = 0 and the PDMA of eofdata_q register is set due to a Data FIS received, it can cause data corruption on the AHB bus during the next initiated transfer (the current transfer is aborted, as expected). It happens if the software clears the CMD.ST bit to send a soft reset while there is still an outstanding data command.

Projected Impact:

Data corruption on the AHB bus (same data is written twice on consecutive addresses).

The severity is low. The probability is low.

Workarounds:

Ensure that there are no outstanding data commands, before clearing the CMD.ST bit.

Proposed Solution:

No fix scheduled

ENGcm11084-3 SATA: Supported AHCI version is incorrect

Description:

The VS.MNR field (Minor Version Number) reset value is 0x0100. The correct value should be 0x0300 indicating support for AHCI 1.3.

Projected Impact:

None. The severity is low.

Workarounds:

None

Proposed Solution:

No fix scheduled

ENGcm11084-4 SATA: DMA Setup FIS with inactive slot reception

Description:

When DMASTP FIS with inactive slot (TAG with corresponding P#SACT bit is zero) is received, it is currently posted to the memory and the data transfer phase is initiated from the wrong slot location. The correct behavior should generate IFS intrq and transition to Fatal state.

Projected Impact:

IFS is not asserted when DMASTP with inactive slot is received, 'bad' DMASTP FIS is posted to the memory, and data transfer operation is initiated resulting in wrong data transfer and likely bus error or other errors (such as incorrect PRD and so forth).

The severity is low. The probability is low.

Workarounds:

None

Proposed Solution:

No fix scheduled

ENGcm11084-5 SATA: Host does not detect disconnect in L_TMPartial/L_TMSlumber states

Description:

When the Device and the Host send PMREQ at the exact same time, but the Device is disconnected before the Host actually detects a PMACK or PMNAK in the deframer, the Host core remains in the power state (L_TPMPartial or L_TPMSLumber) until a COMINIT is received. Also, in this case, the Host core does not flag this as a 'phy not ready' error condition to the Host firmware and the SERR.ERR_C register does not get updated to indicate loss of communication.

Projected Impact:

The Host remains in the L_TPMPartial/L_TPMSlumber state, indefinitely transmitting PMREQ, though there is no Device connected to it, until COMINIT is received from the Device. In addition, the SERR.ERR_C register does not get updated to indicate loss of communication.

The severity is low. The probability is low.

Workarounds:

None

Proposed Solution:

No fix scheduled

ENGcm11084-6 SATA: Possible core lockup after switching from the low power mode

Description:

A problem is identified with the BCM 30 Datastream FIFO, where it has approximately a 1/8 or a 1/16 chance of losing the first data, causing a lock up after a low power mode, due to a problem with the reset logic. When the problem occurs, the FIFO loses the first data that contains alignment information. This results in the core locking on the first non-ALIGN Primitives and the core being locked. The only recovery at this point is a software time-out and the Host issued COMRESET.

Projected Impact:

The Device locks-up after it switches from the low power mode.The severity is high.

Workarounds:

None

Proposed Solution:

No fix scheduled

ENGcm11084-7 SATA: Incorrect interpretation of A-bit in BIST Activate FIS

Description:

As per the SATA specification, in the far-end Tx-only responder mode, the BIST Activate FIS A-bit means ALIGN Bypass (do not transmit ALIGN primitives) when A = 1. The AHCI controller only accepts A = 0 and does not send ALIGNs. A = 1 causes the controller to R_ERR the BIST Activate FIS. This behavior is opposite to what the specification requires.

The correct behavior should be:

- A = 0 - R_ERR
- A = 1 - R_OK (no ALIGNs sent)

Projected Impact:

None

Workarounds:

None

Proposed Solution:

No fix scheduled

ENGcm11084-8 SATA: BIST responder re-timed loopback mode may drop data due to FIFO overflow

Description:

The DWC SATA AHCI core removes and replaces all ALIGNs in the far-end re-timed BIST responder loopback mode. This can cause overflow in the BIST FIFO, if the Tx frequency of the device is faster than the Host Tx frequency (clk_asic). According to the SATA specification, the link can consume up to two ALIGNs and prevent the overflow.

Projected Impact:

The loopback data is corrupted when the BIST FIFO overflows.

The severity is high. The probability is medium.

Workarounds:

Disable the spread spectrum clocking (SSC) on the Host side.

Proposed Solution:

No fix scheduled

ENGcm11084-9 SATA: Host may not wake up from low power mode

Description:

Under certain conditions, the Host controller fails to wake up from the low power mode. The problem arises when the Host issues a COMRESET (software requested reset of system) at the exact time when the Device is disconnected and then reconnected, and when both the Host and the Device were previously in the power down mode. The series of events to cause this error are described as follows:

1. The Host sends a partial power mode request.
2. The Device issues a partial power mode request (collision).
3. The Host backs off and accepts the device power mode request.
4. Both the Host and the Device go into slumber power mode (phy_slumber asserted).
5. The Device is disconnected and reconnected, after which it sends COMINIT.
6. The Host does not respond to COMINIT as it issued a COMRESET from the application software at the exact time COMINIT is detected.
7. The Host does not detect COMINIT, but phy_slumber is de-asserted and asserted again and remains asserted.
8. The Host hangs in the power down mode.

This problem is caused by signals from the clk_asic clock domain remaining asserted after clk_asic is restored, and the signals cause another power mode to be asserted.

Projected Impact:

The Host may not wake up from the low power mode.

The severity is medium. The probability is low.

Workarounds:

The core can come out of this hang condition if software issues another COMRESET.

Additionally, if the Device supports Asynchronous Signal Recovery, after 10ms, the Device reissues COMINIT and OOB completes successfully at that point.

Proposed Solution:

No fix scheduled

ENGcm11084-10 SATA: Injected primitive error inside FIS causes CRC error

Description:

A CRC error is incorrectly asserted for a subsequent FIS after a SYNC primitive is injected inside the incoming FIS. As per the SATA specification, if a SYNCp is received, the other side should not finish the FIS (SATA specification violation). This error condition represents an extremely rare case of data or non SYNCp Primitive being corrupted to be SYNCp, and then the FIS is completed.

Projected Impact:

The Host responds to the subsequent FIS with R_ERR. This error is handled by the system in the usual fashion. For a non-data FIS, the FIS is tried again. For data FIS, the command fails and the error should be handled by the software.

The severity is low. The probability is low.

Workarounds:

None

Proposed Solution:

No fix scheduled

ENGcm11084-11 SATA: SYNC in Rx Data FIS causes PDMA FSM lock up

Description:

If a SYNC is inserted immediately after the first Rx Data FIS DWord (46h header), the PDMA locks up, as it expects the next DWord to be data.

Projected Impact:

The PDMA locks in DR_Receive_prerd state. SERR.DIAG_S bit is set, but interrupt is not generated.

The severity is low. The probability is low.

Workarounds:

COMRESET after time-out.

Proposed Solution:

No fix scheduled

ENGcm11084-12 SATA: Synchronizer misses last wake pulse when CMD.ST bit is cleared in aggressive power mode

Description:

If the P#CMD.ST bit is cleared immediately after the core enters the low power mode using aggressive power management (after the last outstanding command is finished as reported by the CI bits for non-NCQ commands or the SACT bits for NCQ commands), issuing a new command by setting P#CMD.ST and P#CI bits can cause the core to get stuck trying to wake up.

Projected Impact:

The PDMA state machine (portsm_cs) remains in PM_WakeLink state as the three pulses (two acks and one wake) generated are close to each other on the pulse synchronizer (pm_host_ack/wake) causing the synchronizer to miss the last wake pulse. The core does not process any new commands issued.

Under the normal operation, the P#CMD.ST bit is not cleared.

The severity is low. The probability is low.

Workarounds:

Do not clear the P#CMD.ST bit when new commands are issued and the aggressive PM is enabled. Alternatively, issue COMRESET when the command times out when this condition is detected.

Proposed Solution:

No fix scheduled

ENGcm12374 SRTC loses data while powering up/down

Description:

In rare cases the Secured Real Time Counter (SRTC) module can incorrectly invalidate its counters. During a power up or down sequence, the SRTC module supply source is shifted between a coin cell (NVCC_SRTC_POW) and VDDA. The signal that initiates the supply switch from NVCC_SRTC_POW to VDDA is generated based on the output of the LDO regulator which powers up when VDD_REG is applied. If VDDA is not present before VDD_REG, the logic switches to the power supply that's not available yet. When this occurs, the SRTC module may incorrectly report a security violation causing the data (counter and register values) to be lost.

Devices may not function properly due to incorrect toggling of the TEST_MODE input (refer to erratum ENGcm12374). For security reasons, toggling of the TEST_MODE signal causes the register contents of the SRTC to be cleared.

Projected Impact:

- The SRTC module may experience an inadvertent register reset during power up or power down events of the IC. A clock tampering security violation is incorrectly recorded in the SRTC status register.
- The loss of register contents during a power cycle prevents the module from maintaining the time of day functionality.
- Both the secure and non-secure operation of the SRTC module are impacted.
- The SRTC module can still be used for applications that do not require the registers to be preserved over a power cycle.

Workaround:

There is no guaranteed workaround that prevents the SRTC counter reset issue from occurring. Hence it is recommended to use an external RTC solution (PMIC or standalone RTC) to completely avoid the issue. To reduce the likelihood of the SRTC data loss issue, a modified power up/down sequence can be used:

— The NVCC_RESET supply must ramp up along with the VCC supply.

— The power up sequence must bring up the supplies in the following order:

VCC => VDDA => VDD_REG

The ramping supply must be fully powered up and stable per the datasheet before ramping up the next supply in the above sequence.

The power down sequence must shut down the supplies in the following order:

VCC => VDD_REG => VDDA or

VCC => VDDA => VDD_REG

NOTE

The above power sequences reduce the likelihood of the SRTC data loss issue, however does not guarantee that the issue can be completely avoided.

Silicon Fix:

A silicon revision is planned. Please contact your Freescale representative for details.

With revised silicon, the VDDA before VDD_REG power sequencing requirement will remain.

ENGcm11129 SSI: In AC97, 16-bit mode, received data is shifted by four bit locations

Description:

In AC97, 16-bit mode, the Rx data is received in bits [19:4] of RxFIFO, instead of [15:0] bits.

Projected Impact:

The SDMA script should be updated accordingly to perform the shift to the right location on the fly during data transfer. If the data register is accessed directly by software, it should account for the shifted data and perform shifting to the right location.

Workarounds:

The data should be shifted to the right location by the SDMA script or by the software in case of direct access to the register.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required because the AC97 mode is not supported.

WinCE BSP Status:

Not required because the AC97 mode is not supported.

ENGcm11151 USB: Erroneous descriptor handling by USBOH module

Description:

The USBOH core uses the erroneous hrdata (when hresp is high) and results in the false USB transfer at the external Interface.

Projected Impact:

False transfer of USB.

Workarounds:

This should not happen in real system because when configured correctly, the USB should not get an error response. If it happens, then Garbage In Garbage Out.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Not required

WinCE BSP Status:

Not implemented

ENGcm11331 USB: High Speed Transceiverless Logic interface (HS-TLL) and Full Speed Transceiverless Logic interface (FS-TLL) USB interfaces are not supported

Description:

HS-TLL and FS-TLL are not supported. These features will be removed from the specification.

Projected Impact:

Removed support for the option of on board USB connection without transceiver.

Workarounds:

None. Need to add transceiver for on board connection.

Proposed Solution:

No fix scheduled. Feature is removed from the reference manual.

ERR006308 USB: Host controller lock-up issue

Description:

The USB host controller can lock up when a FIFO underrun occurs on a non-32-bit aligned data buffer. This applies to both the Host controller and the OTG controller in Host mode.

Projected Impact:

Ethernet over USB is sensitive to trigger this issue since the data buffers in this case are usually not 32-bit aligned (in Linux).

Workarounds:

1. Set Stream Disable bit (SDIS) in the USBMODE register. This forces the controller to load an entire packet in the FIFO before starting to transmit on the USB bus. Hence, the FIFO never underruns. This somewhat reduces the max bandwidth of the USB since there is idle time as the controller waits for the entire packet to be loaded.
2. Instead of setting SDIS, the FIFO threshold can be increased so that more data is in the FIFO before a packet transmit is started. This increases the tolerance to bus latency and avoids FIFO underrun. The threshold can be increased by using higher values for the TXTHRESHOLD filed in the TXFILLTUNING register. The default value is 2 bursts (64 bytes if burst size=8).

Proposed Solution:

No fix scheduled.

ENGcm10380 VPU: JPEG decoder does not support different AC/DC Huffman tables for Cb and Cr

Description:

There is a limitation when selecting different Huffman tables for AC/DC coefficients for the two chrominance components, Cb and Cr. The JPEG decoder design cannot handle two different Huffman tables for Cb and Cr. Cb and Cr are assumed to use the same Huffman table, which is true in most cases.

Projected Impact:

The bug occurs when the two chroma components, Cb and Cr, use different Huffman tables for AC/DC coefficients. Therefore, if Cb and Cr use the same Huffman table, this problem does not occur. Normally, JPEG streams use the same Huffman tables for Cb and Cr. It should be rare for JPEG streams to use different Huffman tables for the two chroma components.

There is a large visual quality degradation. However, in reality, the probability of using two different Huffman tables for Cb and Cr is low. As a result, the overall impact for JPEG decoding should not be significant.

Workarounds:

None. There is no firmware workaround fix for this erratum.

Proposed Solution:

No fix scheduled

WinCE BSP Status:

Not implemented

ENGcm11195 VPU may miss generating encoding/decoding interrupt when automatic clock gating is activated

Description:

If the CMEOR bit in CCM is not overridden, the VPU clock can be gated off automatically by the vpu_idle signals, while the VPU enters idle state. The VPU may miss generating the decoding/encoding done interrupt when the clock is automatically gated off by the vpu_idle signal.

Projected Impact:

The automatic clock gating can be disabled and replaced by software controlled clock gating with minor impact to power saving.

Workarounds:

Do not activate the automatic clock gating and use the software implementation for VPU clock gating, instead. The VPU clock is gated on before sending a command and gated off after receiving interrupt of the command.

Proposed Solution:

No fix scheduled.

ENGcm11856 IPU and VPU may present some artifacts when running at 200MHz

Description:

Due to a memory timing problem on memories used in the i.MX53 IPU and VPU, these blocks may miscalculate incoming data for decoding. The problem occurs when the IPU and VPU are running at max frequency of 200MHz.

Projected Impact:

Some artifacts may be seen on the decoded image.

Workarounds:

None

Proposed Solution:

Fixed in i.MX53 revision 2.1.

ENGcm12290 eSDHCv3 on eSDHC3 port has setup timing issue in SD SDR mode

Description:

When eSDHC3 port is used in single data rate (SDR) SD mode, it cannot meet the required setup time to operate at a maximum frequency of 50MHz. In this mode it can support up to 45MHz for SD clock frequency. In EMMC4.4 mode, the required setup time is smaller, thus there is no issue and up to 52MHz is supported.

Projected Impact:

For SD cards, which can support 50MHz clock, this frequency may not be achieved on the eSDHC3 port in SDR mode thus affecting bandwidth. Nevertheless, no issue has been reported so far using 50MHz. There is no impact on other SD ports.

Software Workaround:

Use a SD clock frequency up to 45MHz for this port when SD card is used.

Silicon Fix:

No plan to fix.

Linux BSP Status:

No fix implemented.

WinCE BSP Status:

No fix implemented.

ENGcm12360 eSDHC AutoCMD12 and R1b polling problem

Description:

Occurs when a pending command which issues busy is completed. For a command with R1b response, the proper software sequence is to poll the DLA for R1b commands to determine busy state completion. The DLA polling is not working properly for the ESDHC module. This is relevant for all eSDHC ports (eSDHC1-4 ports).

Projected Impact:

DLA bit in PRSSTAT register cannot be polled to wait for busy state completion.

Software Workaround:

Updated block guide to reflect that DLA is not applicable to detect busy state, instead, should poll bit 24 in PRSSTAT register (DLSL[0] bit) to check that wait busy state is over.

Silicon Fix:

No plan to fix.

Linux BSP Status:

Already in ER1109 code base.

WinCE BSP Status:

Already implemented.

ENGcm12354 SCC key fusing with a non-unique value

Description:

The 256-bit SCC fuse (SCC_KEY[255:0]) has been programmed with the same value across i.MX53 devices. The SCC key should be unique per device, to ensure that data encrypted with one i.MX53 cannot be decrypted by another i.MX53.

Projected Impact:

A non-unique SCC_KEY means that the binding of the encrypted data to the IC is no longer unique to that IC. This affects the secure off-chip key storage functionality. Secure off-chip key storage may be used to protect data that is persistent in nature. Secure off-chip key storage is only enabled when the IC has been booted using the secure functionality. If the IC is not securely booted then secure off-chip key storage is not an available feature and thus the IC is not affected by this erratum.

Software Workaround:

None.

Silicon Fix:

No silicon fix required. The unique SCC_Key has been implemented.

Contact factory for additional information.

ENGcm12363 NFC wrong indication of ECC uncorrectable error occurrence after reading the spare area

Description:

In the event that an uncorrectable ECC error occurs while reading Main/Main+Spare from the NAND device, then all spare read operations would fail until the next successful Main/Main+Spare read operation.

Projected Impact:

The NFC wrongly indicates ECC error.

Software Workaround:

Read main or main+spare after first uncorrectable error.

Silicon Fix:

No hardware fix scheduled. This issue will be addressed in the next BSP release.

Linux BSP Status:

No fix is required, as it is not a relevant use-case.

WinCE BSP Status:

Will be ready in 1209 patch release.

ENGcm12377 ESDCTLv2 might fail to wait the minimal time between DDR clk & clk enable

Description:

The DDR2 JEDEC standard requires the DDR clock (SDCLK) to start toggling at least 200 μS before the clock enable (SDCKE) signal rise. For DDR3, the minimum time is 500 μS. In the ESDCTLv2 IP implementation, the actual number that is counted is actually a half CKIL cycle less than described in the eSDCTL chapter in i.MX53 RM. Thus, when programming the recommended value in SDE_to_RST field, the result is a half CKIL cycle less than expected, thus violating the above JEDEC requirement. Note that CKIL is a 32 KHz clock.

Impact:

So far, no issue has been seen.

Software Workaround:

For DDR2, in the RST_to_CKE field (ESDCTL_ESDOR[5:0]), program a value of 10 h, to toggle SDCLK at least 7 CKIL cycles before SDCKE assertion. For DDR3, program a value of 23 h to toggle SDCLK at least 33 CKIL cycles before SDCKE assertion.

Silicon Fix:

No hardware fix is required. The i.MX53 RM and the BSP will be updated accordingly.

ENGcm12379 EIM: AUS mode is non functional for devices larger than 32MB

Description:

When the AUS bit is set, the address lines of the EIM are un-shifted. By default, AUS bit is cleared and address lines are shifted according to port size (8, 16 or 32 bits). Due to an error the address bits 27:24 are shifted when AUS=1. For example, CPU address: 0xBD00_0000 ([A27:20]=1101 0000 becomes: 0xB600_0000 ([A27:20]=0110 0000) on the EIM bus. Since A[27:25] is shifted to [A26:24] and A[23:0] is not shifted. As a result A[24] is missed.

Impact:

If the memory used does not exceed 32 MB, there is no impact.

This mode is related to a unique memory configuration that is not often used. Most systems can work in the default mode (AUS=0). Board designers should connect the EIM address bus without a shift (for example, A0→A0 and A1→A1), while working in AUS=0 mode.

Software Workaround:

- Use the AUS = 0 mode (default) while connecting the address signals without a shift (for example, A0→A0 and A1→A1).
- For AUS=1, for devices larger than 32 MB, need to build a memory map that takes this shifting into consideration and does not include A[24] line.

Silicon Fix:

No hardware fix is scheduled.

Linux BSP Status:

NA.

WinCE BSP Status:

NA.

ENGcm12385 ESAI: Channel misalignment may happen in ESAI transmitter data stream when TIEN bit is zero

Description:

When the TIEN bit is set to low, the ESAI transmitter may misalign the data with the desired channels, regardless of the number of pre-loaded data written to ETDR register. As a result, the audio may be transmitted to the wrong channel, causing a channel mismatch. In case of two channels, the channels could be swapped. If TIEN bit is set to high, and the number of pre-loaded data is equal to the number of channels, the data will be sent to the correct channel.

Projected Impact:

The audio may be transmitted to the wrong channel when TIEN=0.

Workaround:

Set TIEN bit, and write the proper initial words to ETDR register.

Linux BSP Status

The workaround is implemented in Linux BSP release rel_imx_3.0.35_12.09.01

Silicon Fix:

No fix scheduled.

ENGcm12386 ETB may not function properly at 1.2GHz

Description:

The operation of the Embedded Trace Buffer (ETB) is not guaranteed at 1.2 GHz. This issue does not occur on most i.MX53 parts.

Projected Impact:

In some cases, the ETB cannot be used at above 1 GHz, by using the default ARM voltage settings.

Workaround:

- Increase the ARM voltage during debug at 1.2 GHz, maintaining the voltage limits specified in the datasheet.
- Perform the debug at 1 GHz instead of at 1.2 GHz.

Silicon Fix:

No fix scheduled.