TCI6631K2L
Multicore DSP+ARM KeyStone II SOC
Silicon Revision 1.0

Silicon Errata

Literature Number: SPRZ419A
May 2014–Revised June 2015

Contents

List of Figures

List of Tables

TCI6631K2L Multicore DSP+ARM KeyStone II SOC Silicon Revision 1.0

This document describes the silicon updates to the functional specifications for the TCI6631K2L fixed-/floating-point digital signal processor. See the device-specific data manual for more information.

1 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all devices and support tools. Each family member has one of two prefixes: X or [blank]. These prefixes represent evolutionary stages of product development from engineering prototypes through fully qualified production devices/tools.

Device development evolutionary flow:

- **X:** Experimental device that is not necessarily representative of the final device's electrical specifications
- **[Blank]:** Fully qualified production device

Support tool development evolutionary flow:

- **X:** Development-support product that has not yet completed Texas Instruments internal qualification testing.
- **[Blank]:** Fully qualified development-support product

Experimental (X) and fully qualified [Blank] devices and development-support tools are shipped with the following disclaimer:

- ***Developmental product is intended for internal evaluation purposes***.

Fully qualified and production devices and development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that experimental devices (X) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, AAW), the temperature range (for example, blank is the default case temperature range), and the device speed range, in Megahertz (for example, blank is 1000 MHz [1 GHz]).

For device part numbers and further ordering information for TCI6631 in the CMS package type, see the TI website www.ti.com or contact your TI sales representative.

2 Package Symbolization and Revision Identification

The device revision can be determined by the lot trace code marked on the top of the package. The location of the lot trace code for the CMS package is shown in Figure 1. The figure also shows an example of TCI6631 package symbolization.

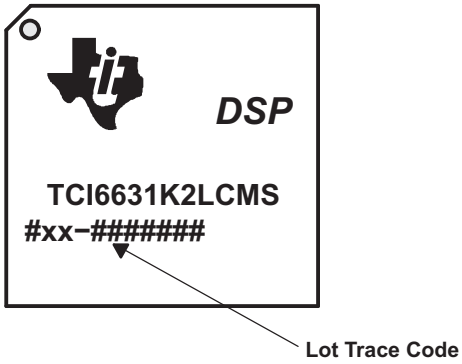

Figure 1. Lot Trace Code Example for TCI6631K2L (CMS Package)

Silicon revision correlates to the lot trace code marked on the package. This code is of the format #xx-#######. Note that there may be an additional leading character (not shown in this example) and xx may actually be two or three characters. If xx is ***10***, then the silicon is revision 1.0. Table 1 lists the silicon revisions associated with each lot trace code for the TCI6631 devices.

Table 1. Lot Trace Codes

Lot Trace Code (xx)	Silicon Revision	Comments
10	1.0	Initial silicon revision

The TCI6631 device contains multiple read-only register fields that report revision values. The JTAG ID (JTAGID) and C66x CorePac Revision ID registers allow the customer to read the current device and CPU level revision of the TCI6631.

The JTAG ID register (JTAGID) is a read-only register that identifies to the customer the JTAG/Device ID.

The C66x CorePac Revision ID register is a read-only register that identifies to the customer the revision of the C66x CorePac. The value in the VERSION field of the C66x CorePac Revision ID Register changes based on the version of the C66x CorePac implemented on the device. More details on the C66x CorePac Revision ID register can be found in the part-specific data manual.

Table 2 shows the contents of the C66x CorePac REVID Register, and the JTAGID register for each silicon revision of the TCI6631 device.

Table 2. Silicon Revision Variables

Silicon Revision	C66x CorePac REVID Register (address location: 0x0181_2000)	TCI6631 JTAGID Register (address location: 0x0262_0018)
1.0	0x0009_ 0003	0x0B9A_602F

More details on the JTAG ID and CorePac Revision ID Registers can be found in the device-specific data manual.

3 ARM-Specific Information

This document does not list the errata for Cortex™-A15 MPCore. For the latest information regard ARM issues, please see the following ARM web pages:

- http://infocenter.arm.com/help/index.jsp
- http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.cortexa/index.html

Table 3 provides the ARM Cortex-A15 MPCore processor version and REVIDR used by the TCI6631.

Table 3. Cortex-A15 Processor Version and REVIDR

SoC	A15 Version	ARM REVIDR
TCI6631K2L	r2p4	0x020A • REVIDR[1] = 1'b • REVIDR[3] = 1'b • REVIDR[9] = 1'b

The ARM product revision $r_m p_n$ indicates the major and minor revision status of the ARM core incorporated in this device. Additionally, for a specific product revision a few additional erratum may be fixed, which can be determined by reading the ARM REVIDR register where a set bit indicates that the erratum is fixed in this ARM revision. A combination of this information can be used when referring to the *ARM Processor Cortex™-A15 MPCore – Product Errata Notice* documentation to infer the erratum applicability to this device.

The definition of the Revision ID Register can be found at the following location:

- http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0438i/CIHEJBDJ.html

4 Silicon Updates

Table 4 lists the silicon updates applicable to each silicon revision. For details on each advisory, click on the link below.

Table 4. Silicon Revision 1.0 Updates

Category	Silicon Update Advisory	See[1]	Applies To Silicon Revision
			1.0
ARM	Execution in Place (XIP) from NOR Flash on ARM Does Not Work	Advisory 2	X
RESET	$\overline{\text{RESETSTAT}}$ Signal Driven High	Advisory 15	X
CCS	System Reset Operation Disconnects SoC from CCS	Advisory 17	X
DDR3	False DDR3 Write ECC Error Reported Under Certain Conditions	Advisory 20	X
PCIe	PCIE MSI/Legacy IRQ Does Not Work for Root Complex	Advisory 22	X
Boot	Boot ROM NAND Cannot Cross Bad Blocks	Advisory 24	X
Boot	ROM Ethernet Boot Failure	Advisory 25	X
IQN	IQN2 MUX Works Only with AIL0 and Not With AIL1	Advisory 26	X
PCIe	Descriptors Placed in PCIe Memory Space can Cause Problems	Advisory 28	X
NSS	NSS Rate Limit Mismatch	Advisory 32	X
Boot	ARM Boot Can Fail When Interrupt Enabled	Usage Note 4	X
Boot	Boot I^2C Frequency Incorrect	Usage Note 5	X
DDR3	Access to DDR3 Without Configuring PHY Properly Can Cause Hang	Usage Note 6	X
Power	Core Wake Up on $\overline{\text{RESET}}$	Usage Note 9	X
I^2C	I^2C Bus Hang After Master Reset	Usage Note 10	X
PLL	Minimizing Main PLL Jitter	Usage Note 11	X
QMSS	Queue Proxy Access	Usage Note 14	X
Power	Initial Voltage Level Setting of CVDD Rail Power Supplies	Usage Note 17	X
USB	USB Hangs When Doing a Master Access to Reserved Space	Usage Note 25	X
SerDes	SerDes Fails to Adapt RX BOOST Equalization	Usage Note 28	X

(1) Not all KeyStone II errata apply to all KeyStone II parts. Therefore, numbering gaps in the errata list are normal.

Silicon Updates

Title **Page**

KeyStoneII.BTS_errata_advisory.2

Execution in Place (XIP) from NOR Flash on ARM Does Not Work

Revision(s) Affected 1.0

Details ARM cannot perform direct execution from a parallel NOR flash connected via ASYNC EMIF.

On these devices, ARM cannot perform direct execution from a parallel NOR flash connected via ASYNC EMIF. Direct execution from a parallel NOR flash does not work as ARM always generates 64-byte cacheline wrap mode accesses to EMIF. This is irrespective of marking this memory region as Device or Strongly Ordered as confirmed by ARM. ASYNC EMIF does not support 64-byte cacheline wrap accesses and therefore generates a bus error on receiving it. This causes an abort to happen in ARM. The ARM is, however, able to read data from a parallel NOR flash connected via ASYNC EMIF.

Workaround None. If code is stored on NOR flash, it must be copied from the NOR to another memory area and executed from there.

KeyStoneII.BTS_errata_advisory.15

$\overline{\text{RESETSTAT}}$ Signal Driven High Issue

Revision(s) Affected 1.0

Details The $\overline{\text{RESETSTAT}}$ output signal should be driven low when a reset is applied and held low until the reset cycle is complete. If the device is using power sequencing where the 1.8 V (DVDD18) is present before the AVS core voltage (CVDD), the $\overline{\text{RESETSTAT}}$ signal may be driven high erroneously during the time between when DVDD18 is present and the CVDD is present.

Workaround One workaround is to use the CVDD before DVDD18 in the power sequencing. An alternative workaround is to ignore the $\overline{\text{RESETSTAT}}$ signal if the CVDD is not present during the power sequencing.

KeyStoneII.BTS_errata_advisory.17

System Reset Operation Disconnects SoC from CCS Issue

Revision(s) Affected 1.0

Details CCS connection to targets will fail after system reset issued via CCS. CCS connection to targets will also fail after $\overline{\text{RESET}}$ reset of the device. A system reset, issued from CCS or by the $\overline{\text{RESET}}$ pin, can cause power reset to all C66x Corepacs and can cause the hardware states of debug logic (including hardware breakpoints) to get cleared. The result is that any existing CCS connection to those targets will get corrupted, terminating further access to the target.

Workaround 1: A new configuration option called Domain Power Loss Mode is added in the CCS target configuration for enabling the debug software to detect and handle the power loss event automatically.

To enable this option, in the CCS target configuration window, click on the sub-path of ICEPICK_D for each individual C66x Corepac. Then click on the property option **Domain Power Loss Mode** and select **Auto**.

The support for this new option will be released in the emupack update v5.0.586.0 or newer, patched to CCS5.1 GA.

Workaround 2: Before issuing a system reset, disconnect CCS from all DSP targets, issue the system reset, then reconnect CCS to the targets to continue debug operations.

KeyStoneII.BTS_errata_advisory.20

False DDR3 Write ECC Error Reported Under Certain Conditions

Revision(s) Affected 1.0

Problem Summary: In the event that the read-modify-write (RMW) ECC feature for DDR3 is not supported or disabled, an L1D or L2 block writeback or writeback invalidate operation to ECC protected DDR3 space will flag a DDR3 write ECC error, even though neither the data nor the ECC values stored in the SDRAM will be corrupted.

Details The write ECC error interrupt can be enabled by setting the WR_ECC_ERR_SYS bit in the Interrupt Enable Set Register (IRQSTATUS_ SET_SYS) of the DDR3 controller.

Under normal conditions, a write access performed within the ECC protected address range to a 64-bit aligned address with a byte count that is 64-bit quanta is not expected to flag a write ECC error interrupt. The C66x cache controller always operates on whole cache lines, which are 128 bytes for the L2 cache and 64 bytes for L1D cache. However, a block writeback or writeback invalidate always generates a bounding single byte write (with its byte enables disabled) to the last address in that block. This single byte write violates both the alignment and quanta conditions causing the DDR3 controller to flag a write ECC error in the Interrupt Raw Status Register (IRQSTATUS_RAW_SYS) register. Since this bounding write is sent with its byte enables disabled, it does not actually reach the DDR memory and does not corrupt the stored data or ECC values. The write ECC error is thus a spurious error since no data or ECC value is actually corrupted. It should be noted that the DDR3 controller, in response to this sub-quanta write (the bounding single byte write), will report an error on an internal status line to the CPU that executed the writeback. This error status flags the MDMA error interrupt to the CPU and is interpreted as an MDMA data error (the STAT field in the CPU's MDMA Bus Error Register will be set to 0x4).

NOTE: 1: Since no bounding writes are generated with global writeback or global writeback invalidate operations, this issue is limited only to block writebacks to ECC protected region.

NOTE: 2: If the MDMA error flagged by a block coherence operation is followed by a true MDMA error flagged by a master executing a direct sub-quanta write, only the first MDMA error will be captured. Software must clear an MDMA error as soon as possible in order for future errors to be captured.

Workaround 1 The problem can be resolved by enabling the read-modify-write ECC feature for DDR3 (set RMW_EN=1 in the ECCCTL register of the DDR3 memory controller). The RMW feature is specifically designed to handle sub-quanta/non-aligned accesses to ECC protected space. With RMW enabled the bounding single byte write will not violate any quanta/alignment requirements and thus will not flag a write ECC error.

This device supports RMW.

Workarounds 2 and 3 should be explored if RMW is not available or not used.

Workaround 2 In order to differentiate a false write ECC error from a true error generated by alignment/quanta violations, the system should keep track of block writeback/writeback invalidate operations to the ECC protected memory space. If a write ECC error is confirmed for that operation, it can be safely ignored. There is only a single DDR3 error interrupt that will have to be processed by one of the C66x cores. Therefore, some special mechanism will be required for the system to keep track of which core performed the block writeback that caused the error. This mechanism may involve checking for the data error reported in the MDMA Bus Error Register.

NOTE: 3: A system must satisfy the alignment/quanta conditions so a true write ECC error is not expected and such errors should be isolated and removed as part of system software evaluation.

NOTE: 4: The system software must clear the write ECC error and MDMA error before they can be re-triggered by any successive error conditions. It should be noted that a race condition can exist if a subsequent ECC error (real or false) or MDMA error interrupt is triggered before the previous interrupt is cleared.

Workaround 3 A global coherence operation can be performed instead of a block operation. It should be noted that a global operation can possibly operate on more cache lines than the block operation, causing a larger than necessary cycle overhead and negatively impact memory system performance.

FAQ: Q: The C66x CorePac will receive an MDMA error in response to the DDR3 ECC error. Other masters may also see the DDR3 ECC error when transactions they have sent result in the error. How do these masters respond to a DDR3 ECC error?

A: The responses of the C66x CorePac, ARM CorePac, and other masters to the non-zero values returned on rstatus and sstatus due to DDR3 ECC errors are summarized in Table 5.

Table 5. Master Behavior in Response to sstatus/rstatus Flagged Due to DDR3 ECC Errors

Master	Bus Error Returned	Error Status Captured	Alarm Notification
C66x CorePac	sstatus, rstatus	Error status captured in the STAT field in the C66x CorePac's MDMA Bus Error Register will be set to 0x4.	The C66x CorePac's MDMA error interrupt is asserted.
ARM CorePac	sstatus, rstatus	Details on the exception are captured in the CP15 registers: Data Fault Status Register (DFSR), Instruction Fault Status Register (IFSR) or Auxiliary Data Fault Status Register (ADFSR). The address that generated the abort can be seen by reading Data Fault Address Register (DFAR) for synchronous aborts.	The ARM CorePac will trigger an external abort exception. They are disabled by default and should be enabled via the CPSR.A bit (Current Program Status Register).
PCIe	sstatus, rstatus	Interrupts are not generated and error status is not logged.	PCIe returns completion abort to requestor only for rstatus error. No completion abort is returned for a write error (sstatus).
EDMA TC	sstatus, rstatus	BUSERR and ERRDET registers capture the error information. The transfer of data from source to destination happens irrespective of error.	Error interrupt is generated if enabled within EDMA.
Multicore Navigator Infrastructure PktDMA	sstatus, rstatus	Error status is not logged.	Error interrupt is not reported
TSIP	sstatus, rstatus	Errors will be stored in the channels' interrupt queue along with the error codes.	TSIP asserts an error event (TSIPx_ERRINTn) when an error is queued for channel 'n'. CorePac[n] will receive TSIPx_ERRINTn.
SRIO	sstatus, rstatus	Error responses set a bit in the AMU_INT_ICSR register based on the CPRIVID of the transactions. The RIO_AMU_ERR_CAPT0, RIO_AMU_ERR_CAPT1 registers will contain the address of the non-posted transactions that failed along with the CPRIVID and CMSTID.	Each bit can be routed by software configuration through the Interrupt Condition Routing Register (ICRR) to a specific ARM or C66x CorePac for error handling. See the device data manual for interrupt mapping.

Table 5. Master Behavior in Response to sstatus/rstatus Flagged Due to DDR3 ECC Errors (continued)

Master	Bus Error Returned	Error Status Captured	Alarm Notification
HyperLink	sstatus, rstatus	When the serial link is active HyperLink will pass the rstatus. Rstatus is will be that of the remote slave read.	HyperLink Error interrupt (HyperLink_INT or VUSR_INT) are generated when error is received and are provided to CorePacs as secondary interrupts.
10GbE	sstatus, rstatus are not used	NA	NA

NOTE: Check your device data manual to see which masters are applicable.

KeyStoneII.BTS_errata_advisory.22

PCIE MSI/Legacy IRQ Does Not work for Root Complex

Revision(s) Affected 1.0

Details While testing the AER PCIE error handling and error recovery Linux driver software on KeyStone II EVM, it was found that only error interrupt #12 is raised. An unsupported request error is simulated by generating transaction to an EP with an address not in the BAR. The error is detected by Root complex.

As per PCIE spec 2.0, section 6.2.6, MSI/Legacy error interrupt to be raised as well platform specific interrupt. Currently, only platform specific interrupt (INT #12) is raised. AER driver depends on Legacy/MSI IRQ and can't function without this. This is the standard PCI driver in Linux to handle Error and do recovery.

Workaround None

KeyStoneII.BTS_errata_advisory.24

Boot ROM NAND Cannot Cross Bad Blocks

Revision(s) Affected 1.0

Details

NAND <ARM> boot fails if boot image crosses a bad block boundary.

During a NAND primary boot, the boot ROM will read and process an image from the NAND device specified by the bootstrap pins. When the boot ROM encounters a pre-marked bad block or detects one through error correction, data processing will reset and invalidate previously read good data. This is opposite U-Boot NAND reads which skip bad blocks and continue reading data.

In other words, if a primary boot image (such as U-Boot) spans multiple blocks with a bad block separating valid data, the boot ROM will not read the complete boot image. Instead it will read only the data following a bad block. Boot will ultimately fail without a complete boot image.

Table 6. Proper way to write boot image starting with Data 0

Block # :	Block 0	Block 1(Bad)	Block 2	Block 3	Block 4
Data # :	XXXXX	XXXXX	Data 0	Data 1	Data 2
Result :	Skipped	Skipped	^ Data 0 will be the start of the image.		

Table 7. Do not write boot image this way

Block # :	Block 0	Block 1(Bad)	Block 2	Block 3	Block 4
Data # :	Data 0	XXXXX	Data 1	Data 2	Data 3
Result :	Thrown Away	Skipped	^ Data 1 will now be the start of the image.		

Workaround

If the boot image is large enough to require multiple blocks, locate a sufficient amount of consecutive good blocks to store boot image. Tools such as U-Boot have built-in commands to list bad NAND block as well as the ability to write to a given offset.

This issue does not affect a 2nd stage boot to Linux from U-Boot; so the kernel and filesystem may be written to skip bad blocks as per default behavior. Default U-Boot NAND reads will skip bad blocks and continue reading data.

If NAND boot is used as primary boot to boot U-Boot (or boot image) then, ‘NAND writer utility’ which writes U-Boot (or boot image) to NAND needs to understand this errata and make sure that U-Boot (or boot image) is burned in consecutive blocks with no bad blocks in the middle. In some cases, the ‘NAND writer utility’ could be U-Boot itself then customer U-Boot needs to modify to incorporate this errata workaround.

KeyStoneII.BTS_errata_advisory.25

ROM Ethernet Boot Failure

Revision(s) Affected 1.0

Details

As per Ethernet boot sequence, the booting device sends BOOTP packets out. In response, the boot master sends the BOOTP response packet. In the failing scenario, the BOOTP packets will be seen exiting the device, but the BOOTP response appears to be ignored by booting device.

The root cause is identified as an uninitialized value in the Ethernet driver in the BootROM code. Packets that arrive to the device are held up in the Ethernet and not passed to the BootROM code for processing. This happens all times when Ethernet boot is selected as primary boot.

This issue is applicable to Ethernet boot for both the C66x CorePac as the boot master and the ARM CorePac as the boot master, please refer to the device specific datasheets for the boot master and boot modes supported.

Workaround

The only workaround is to use a different boot mode (other than Ethernet Boot) as the Primary Boot and then perform the Ethernet Boot. The Primary Boot can be used to initialize the above mentioned uninitialized value and then re-enter the Ethernet Boot as Secondary Boot mode.

Step 1. Enable NETCP.

Step 2. Initialize the registers:

```
#define PA_BASE 0x24000000
ROM Errata workaround Function:

memset ((unsigned int *)(PA_BASE + 0x408400), 0, 32 * sizeof(unsigned int));
memset ((unsigned int *)(PA_BASE + 0x418400), 0, 32 * sizeof(unsigned int));

*((unsigned int *)(PA_BASE + 0x409814)) = 0x10;
*((unsigned int *)(PA_BASE + 0x409820)) = 0x4000;
*((unsigned int *)(PA_BASE + 0x409824)) = 0xfffc0000;
*((unsigned int *)(PA_BASE + 0x409810)) = 0x80000600;
```

Step 3. Write the Ethernet boot mode configuration to the DEVSTAT register.

Step 4. Branch to re-enter the ROM boot function.

KeyStoneII.BTS_errata_advisory.26

IQN2 MUX Works Only with AIL0 and Not With AIL1

Revision(s) Affected 1.0

Details IQN2 has two AIL lanes (AIL0 and AIL1). AIL1 lane does not transmit or receive any data. AIL0 lane can transmit or receive data without showing any problem.

As shown in Figure 2 below, the AIL0 and AIL1 lanes are connected to a SERDES MUX with respect DFE lanes. This SERDES MUX is used to select data path between IQN2 and DFE. The mux select for the AIL1 and DFE1 mux is not connected correctly and is connected to Ground. So DFE1 lane is always selected irrespective of the configuration selected by mux pins.

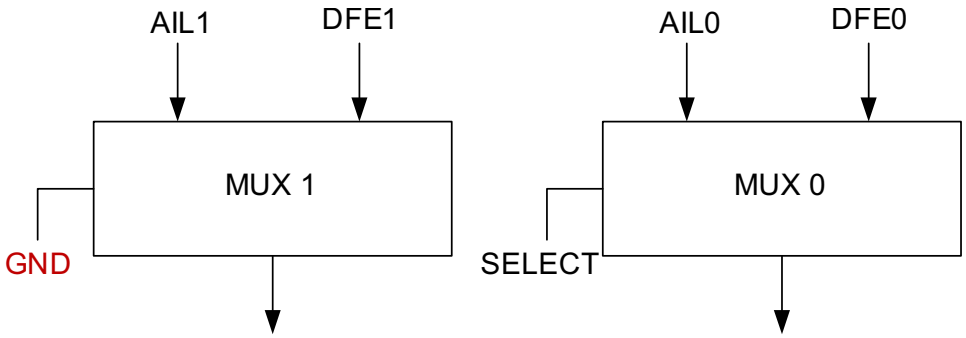

Figure 2. AIL Lanes

Workaround There is no SW workaround for this problem.

KeyStoneII.BTS_errata_advisory.28

Descriptors Placed in PCIe Memory Space can Cause Problems

Revision(s) Affected 1.0

Problem Summary: Packet DMA can generate write transactions with partial byte enables when trying to access descriptors. This can cause problems if the descriptors are stored in PCIe memory space since PCIe cannot handle partial byte enables. Here, *partial byte enables* means that the transactions are accessing memory that is not a full 32-bit word.

Details Packet DMA can sometimes generate write transactions with partial byte enables asserted when trying to access descriptors. This can cause problems if the descriptors are placed in PCIe memory space since PCIe does not support transaction requests with partial byte enables. Such a transaction will corrupt the PCIe module's internal state machine and in turn corrupt the payload. This issue does not impact Packet DMA access to data buffers, only to descriptors since all byte enables are asserted in transactions involving data buffers. Thus, the data buffers may be stored anywhere including in PCIe memory space.

Workaround As long as host-mode descriptors are used and these descriptors are located in a memory space that can properly handle partial byte enables (such as L2 SRAM, DDR3 or MSMC), the issue will not affect Packet DMA accesses to PCIe memory space. As mentioned earlier, data buffers can be stored in PCIe memory space without any problems.

KeyStoneII.BTS_errata_advisory.32

NSS Rate Limit Mismatch

Revision(s) Affected 1.0

Problem Summary: It has been observed that the Rate Limit block allows more traffic than desired for small size packets. In a particular test case, the rate limit is set to 100Mbps and the CPU is trying to generate 150Mbps of traffic comprised of 60-byte packets. The observed bandwidth of packets received will be 116Mbps.

Details The following equation expresses the relationship between the intended frame-size and the associated rate-error:

Rate error=((((FrameSize)%16)==0)?(16): (FrameSize%16))/(FrameSize+24)

This results in the possibility of error rate as high as 10% with small packet sizes around 64 bytes that diminishes to less than 2% as packet sizes exceed 1K bytes. This can be seen in the graph below:

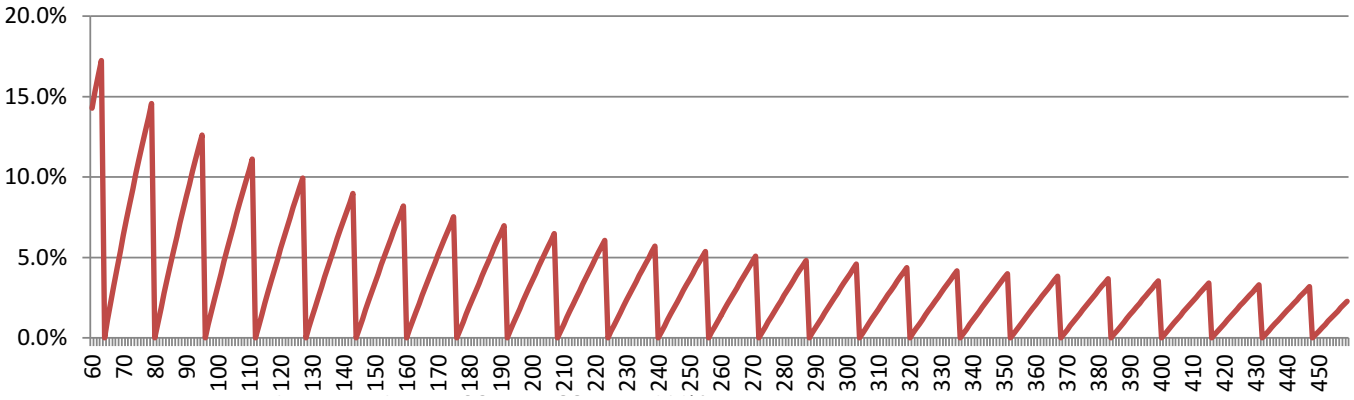

Figure 3. NSS Rate Limit Error % For Small Packets

Workaround Since the rate-error is known, at least for fixed packet sizes, a software workaround can be implemented by calculating the “effective” packet-size to compensate for the calculation error in hardware. It is not possible to establish this workaround for streams with varying packet-sizes. It will only be meaningful when it is known that a port will mostly carry short length packets.

Illustrating with real packet size of 60 (+24 = 84, on the wire), though 84 bytes are sent out on the wire, the rate-limiter’s calculation accounts for only 72 (i.e, 60%16 + 24 = 72) bytes. Therefore, one needs to calculate the “compensated bandwidth” based on the real bandwidth calculated from the effective packet size using the error formula internally. For example, BW_out = BW_in * (84/72).

Once such an ‘error-compensated throughput’ is calculated, the appropriate Mbps value is used to derive the rate-limiter’s configuration values (depending on the formula appropriate to the platform):

Priority Transfer rate in Mbit/s = p0_priX_idle / (p0_priX_idle + p0_priX_send) * frequency * 256

KeyStoneII.BTS_errata_usagenote.4

ARM Boot Can Fail When Interrupt Enabled

Revision(s) Affected 1.0

Details ARM boot can fail when an interrupt is pending when interrupts are enabled.

From the ARM Boot ROM code, the “Enable interrupts” code below in the ROM is not implementing correctly.

```
*******************************************************************************
* Enable interrupts
*******************************************************************************

.def _chipEnableInts

_chipEnableInts:

cpsie i
bx  lr
```

If an interrupt is pending when the cpsie instruction executes the interrupt is immediately taken. The interrupt code executes normally, and even returns with the correct mode. However the next instruction, bx lr, does not execute. The code simply continues into the next instruction, which happens to be a data value. This results in an invalid instruction exception.

For devices including the C66x CorePac:

In a case of the C66x boot master and a $\overline{\text{RESET}}$ is applied, the $\overline{\text{RESET}}$ resets the ARM PLL causing the ARM boot code to execute very slowly. The system PLL was reset isolated and executed very quickly. The C66x boot code was loaded, and this code poked the IPC interrupt to wake up the ARM, but the slow ARM was still setting up translation tables. If there is an interrupt pending, the interrupt is taken when the interrupts were enabled.

This could also happen when the ARM is the boot master in the PCIe and Hyperlink boot modes. If the remote end generates an interrupt immediately after link detection it is possible that the ARM code has not yet reached the cpsie instruction. But in both of these modes, the ARM PLL will be enabled and the race is very short.

The consequence of this is that the ARM generates an invalid instruction exception.

Workaround Delay interrupts to the ARM for a few ms. This applies mostly to ARM core 0. Secondary ARM cores are usually woken up without an interrupt, however the same code can execute if the secondary ARM core wakes up and does not see a branch address, and in which case it enables interrupts and idles.

KeyStoneII.BTS_errata_usagenote.5

Boot I²C Frequency Incorrect

Revision(s) Affected 1.0

Details The issue is within the Boot ROM. Initial boot I²C frequency incorrect on $\overline{\text{RESET}}$ reset.

For I²C boot, the code to determine the device frequency assumes that the PLL is in bypass. This is true for power on reset, but for $\overline{\text{RESET}}$, with the PLL reset isolated this is incorrect. The code should check to see if the PLL is enabled and if it is, it should return the e-fuse device frequency.

On a normal $\overline{\text{POR}}$ or $\overline{\text{RESETFULL}}$, boot with I²C as the boot master, the boot code will correctly assume the system is running at 312 MHz (max possible) and scale the I²C clock to run at 20 KHz. So the actual frequency will scale based on the actual reference clock. After boot execute a $\overline{\text{RESET}}$, the initial frequency will be much higher, running faster by a multiple equal to the actual effective PLL multiplier value.

For example if the actual reference clock is 50 MHz and the device frequency is e-fused for 1400 MHz, the initial I²C will read using a data clock of 20 * 50 / 312 = 3.2 KHz. After a $\overline{\text{RESET}}$ reset, with the PLL reset isolated, the initial read will be at 20 * 1400 / 312 = 89 KHz. This will work with almost all I²C devices that are compliant to the 100 KHz bus standard specified in the original I²C specification. This represents the worst case for these devices.

Workaround None.

KeyStoneII.BTS_errata_usagenote.6

Access to DDR3 Without Configuring PHY Properly Can Cause Hang

Revision(s) Affected 1.0

Details If the DDR3 is not configured properly before the CorePac issues an access to DRR3, the device could lock up.

If the DDR3 PHY Utility Block (PUB), DDR3 PHY and the EMIF Controller are not configured or improperly configured, any access to the DRR3 memory space is issued by the CorePac, including opening a memory window view from the Code Composer Studio (CCS) that is pointed to the DDR3 memory space, the device could lock up.

Workaround It is recommended that before issuing an access to the DDR3, the device must properly initialize the DDR3 PUB, DDR3 PHY, and EMIF controller.

Refer to the KeyStone II DDR3 Programming Sequence documented in the KeyStone II DDR3 User Guide (SPRUGV8) or the KeyStone II DDR3 Initialization Sequence document (SPRABL2).

KeyStoneII.BTS_errata_usagenote.9

Core Wake Up on RESET Usage Note

Revision(s) Affected 1.0

Details

Execution may start only on some CorePacs if CCS is connected to the device and reset is applied via the RESET pin on the device. In order to make sure that all the CorePacs wake up after reset via RESET pin, the device needs to be completely disconnected from the CCS before applying reset via RESET pin.

Some of the CorePacs do not wake up on RESET reset when the device is connected via CCS. When the device is connected via CCS the device stays in the emulation debug state. If the RESET reset is applied while the device is in the emulation debug state it causes some of the CorePacs to go into an unknown state and they don't start execution.

Resets using POR and RESETFULL do not exhibit this behavior.

This does not affect the normal usage of the device when CCS/emulator is not connected to the device since the device is not in emulation debug state when reset is applied using the RESET pin. This behavior can only happen in the lab environment where the CCS/emulator is connected to the device.

Workaround

Below is the sequence which must be followed to completely disconnect the device from CCS before applying RESET:

1. "Free Run" all the CorePacs
2. Disconnect all the CorePacs from CCS
3. Apply RESET

Steps 1 and 2 insure that all the debug states are cleared in the device. This will allow the CorePacs to wake up correctly on reset via RESET. Bypassing either step 1 or 2 will result in CorePacs that do not begin execution after reset.

KeyStoneII.BTS_errata_usagenote.10

I²C Bus Hang After Master Reset Usage Note

Revision(s) Affected 1.0

Details It is generally known that the I²C bus can hang if an I²C master is removed from the bus in the middle of a data read. This can occur because the I²C protocol does not mandate a minimum clock rate. Therefore, if a master is reset in the middle of a read while a slave is driving the data line low, the slave will continue driving the data line low while it waits for the next clock edge. This prevents bus masters from initiating transfers. If this condition is detected, the following three steps will clear the bus hang condition:

1. An I²C master must generate up to 9 clock cycles.
2. After each clock cycle, the data pin must be observed to determine whether it has gone high while the clock is high.
3. As soon as the data pin is observed high, the master can initiate a start condition.

KeyStoneII.BTS_errata_usagenote.11

Minimizing Main PLL Jitter Usage Note

Revision(s) Affected 1.0

Details Once the boot is complete, it is highly recommended that software reconfigure the Main PLL to the desired frequency, even if it is already achieved by the initial settings. To minimize the overall output jitter, the PLLs should be operated as close as possible to the maximum operating frequency. To maximize the VCO frequency within the PLL, the PLL should be clocked to 2× the intended frequency and the PLL Output Divider should be set to /2. The main PLL Output Divider should be set to divide-by-2 by the software by writing 0b0001 to bits [22:19] of the SECCTL register (address 0x02310108) in the PLL controller. A read-modify-write can be used to make sure other bits in the register are not affected. This register is documented in the part-specific data manual.

NOTE: It is only after programming the SECCTL register to enable the divide-by-2 that the following equation can used to program the PLL as specified in the data manual.

CLK = CLKIN × ((PLLM+1) ÷ ((OUTPUT_DIVIDE+1) × (PLLD+1)))

KeyStoneII.BTS_errata_usagenote.14

Queue Proxy Access Usage Note

Revision(s) Affected 1.0

Details When there are multiple DSP cores potentially accessing the Queue Manager, the Queue N register A, B, C, and D should be accessed in the same burst. However, the C66x CorePac cannot generate bursts larger than 8 bytes. The Queue Proxy is designed to allow the C66x CorePac to push/pop descriptors using multiple transactions. However, when the C66x CorePac uses the Queue Proxy region for push and pop, the Queue Proxy may mix the transactions from non-CorePac system masters. This may lead to an error transaction, which causes a system deadlock.

Workaround The C66x CorePac should not use the Queue Proxy region to push/pop descriptors. The C66x CorePac should use VBUSM region (base address starts from 0x34000000) to push descriptors. When Queue N register C is needed for a push, the C66x CorePac should issue a DoubleWord write to generate an 8-byte burst write to Queue N register C and Queue N register D. When device is little endian mode, the Queue N register C and Queue N register D value need to be swapped. The C66x CorePac should use the VBUSP region (base address starts from 0x02A00000) to pop Queue N register D only. When packet size, byte count, and queue size information are needed for a certain queue, C66x CorePac should use the queue peek region to get the information.

KeyStoneII.BTS_errata_usagenote.17

Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note

Revision(s) Affected 1.0

Details Users are required to program their board CVDD supply initial value to 1.0 V on the device. The initial CVDD voltage at power-on will be 1.0 V nominal and it must transition to VID set value, immediately after being presented on the VCNTL pins. This is required to maintain full power functionality and reliability targets guaranteed by TI.

SmartReflex voltage scheme as defined by the device specific data manual and *Hardware Design Guide for KeyStone II Devices* is required.

KeyStoneII.BTS_errata_usagenote.25

USB Hangs When Doing a Master Access to Reserved Space

Revision(s) Affected 1.0

Details When accessing the reserved space using USB AXI on HOST side (by providing the reserved address value to Device Context Base Address Array Pointer) and Host does not provide any interrupt to SW, however it sets the bus_error status flag in USBSTS and GSTS. Hence the USB HOST cannot convey AXI Bus error directly to SW. SW has to make sure that if it does not get the proper response from the USB HOST, it should check the USBSTS/GSTS registers for possible error. On Device side, events generated by USB Device after data transfer, has information about AXI bus error.

Workaround Avoid accessing to reserved space.

KeyStoneII.BTS_errata_usagenote.28

SerDes Fails to Adapt RX BOOST Equalization

Revision(s) Affected 1.0

This problem impacts the following high speed interfaces on all current Keystone-II devices. Refer to the device data manual for applicability. It does not impact 10GbE operation.

Table 8. SerDes peripherals impacted by RX Boost equalization problem

Interface	Data Rate	Notes
AIF2	>4.9Gbps	8b/10b symbols are used during data transport. There is no link training
AIL	>4.9Gbps	8b/10b symbols are used during data transport. There is no link training
Hyperlink	>6.25Gbps	8b/10b symbols are used on Hyperlink only during link training
JESD	>4.9Gbps	8b/10b symbols are used during data transport. There is no link training
PCIe	5Gbps	8b/10b symbols are used during data transport. There is no link training
SRIO	5Gbps	8b/10b symbols are used during data transport. There is no link training

Problem Summary: The SerDes on some peripherals will not automatically adapt its RX equalization when provided with certain data encodings common to that peripheral's electrical standard.

Details The TI SerDes contains an adaptation algorithm that allows the RX equalization blocks to adapt their parameters to the SerDes data channel. The adaptation algorithm sets the optimal values for the ATT, BOOST, and DFE blocks in order to equalize the signal, maximize margin, and minimize channel distortion. For higher data rates (i.e. 5Gbps and greater), the high frequency gain provided by the BOOST block is generally considered necessary to compensate for the low-pass characteristics of data channels and widen the data eye.

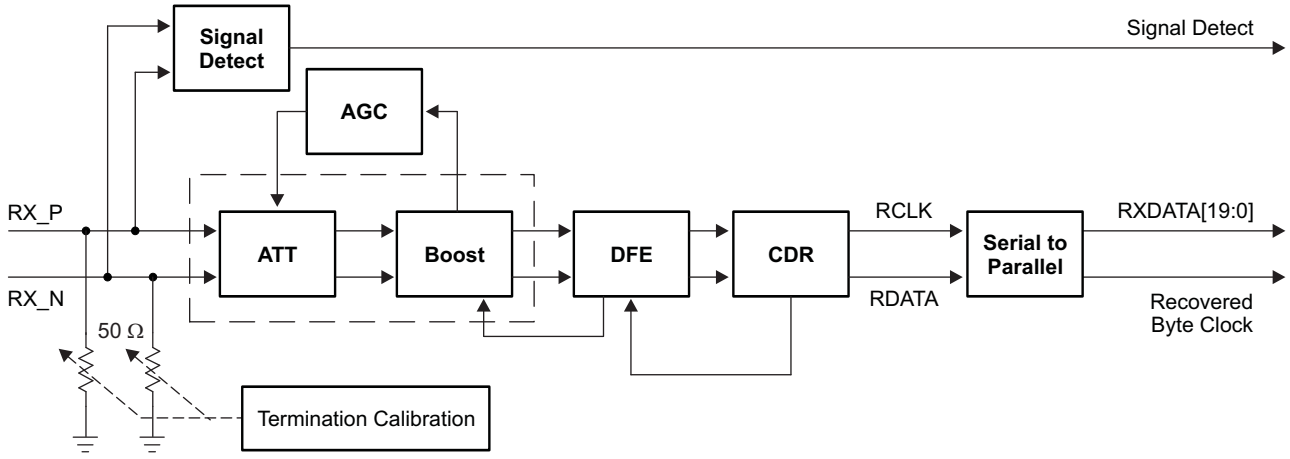

Figure 4. RX Path Block Diagram

The SerDes BOOST block was designed to require a specific set of patterns in order to best adapt its parameters to the channel. If these patterns are not found, the BOOST adaptation algorithm will not update the internal block settings and/or may incorrectly update these settings. The required data pattern is a series of six 0'-s followed by a 101 pattern. This pattern must occur numerous times on both odd and even bit alignments. This data pattern is not present in the 8b/10b coded symbols that are sent over many of the high speed peripheral links.

The data pattern is present in longer length PRBS patterns such as PRBS-23, and PRBS-31 used by the SerDes internal BIST hardware used during BER testing. The data pattern that is required is not present in the symbols used in initial Hyperlink training or in the other high speed SerDes interfaces during data transport.

If the BOOST has not been properly adapted for the received signal, then there is a chance that the SerDes will be unable to recover the RX data or the error rate may be higher than expected.

Workaround

The recommended workaround is to not use the RX equalization auto-adaptation mechanism with the impacted interfaces at higher data rates. As an alternative, a user can hardcode the optimal ATT and BOOST values for their channel.

These optimal ATT and BOOST values must be determined by the user through one of two options:

- Perform BER test with PRBS sequence and sweep across all values of ATT/BOOST while using a fixed set of TX parameters that have already been optimized. Use optimal ATT and BOOST value permutation that has the most margin. This is the value permutation that is both error-free and “farthest” from permutations with errors.
  - The SerDes DIAG tests can be found under *<TI_PDK_INSTALL_DIR>\packages\ti\diag\serdes_diag* in the latest CSL/PDK release and can be configured to perform this test.
- Perform a test where a PRBS data pattern is presented to the RX and the ATT/BOOST are allowed to auto-adapt. This method can also be used to identify the optimal ATT and BOOST value permutation that has the most margin.

The optimal ATT and BOOST values found by one of the two above methods can be hardcoded into the SerDes upon initialization.

Revision History

Changes from June 5, 2014 to May 30, 2015 (from * Revision (June 2014) to A Revision) **Page**

- Added KeyStoneII.BTS_errata_advisory.20, ***False DDR3 Write ECC Error Reported Under Certain Conditions*** 12
- Updated NOTE 2 KeyStoneII.BTS_errata_advisory.20, ***False DDR3 Write ECC Error Reported Under Certain Conditions*** 12
- Updated Workaround 1 KeyStoneII.BTS_errata_advisory.20, ***False DDR3 Write ECC Error Reported Under Certain Conditions*** 12
- Updated Workaround 3 FAQ KeyStoneII.BTS_errata_advisory.20, ***False DDR3 Write ECC Error Reported Under Certain Conditions*** 13
- Added KeyStoneII.BTS_errata_advisory.28, ***Descriptors Placed in PCIe Memory Space can Cause Problems*** 19
- Added explanation for partial byte KeyStoneII.BTS_errata_advisory.28, ***Descriptors Placed in PCIe Memory Space can Cause Problems*** in the Problem Summary 19
- Added KeyStoneII.BTS_errata_advisory.32, ***NSS Rate Limit Mismatch*** 20
- Added KeyStoneII.BTS_errata_usagenote.28, ***SerDes Fails to Adapt RX BOOST Equalization*** 30
- Updated Notes KeyStoneII.BTS_errata_usagenote.28, ***SerDes peripherals impacted by RX Boost equalization problem*** 30
- Updated Details and Workaround KeyStoneII.BTS_errata_usagenote.28, ***SerDes Fails to Adapt RX BOOST Equalization*** 31

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265