TMS320C6672
Multicore Fixed and Floating-Point Digital Signal Processor
Silicon Revision 1.0, 2.0

Silicon Errata

Literature Number: SPRZ335H
January 2011–Revised June 2015

Contents

List of Figures

List of Tables

TMS320C6672 Multicore Fixed and Floating-Point Digital Signal Processor Silicon Revision 1.0, 2.0

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320C6672 fixed-/floating-point digital signal processor. See the device-specific data manual,*TMS320*C6672 *Multicore Fixed and Floating-Point Digital Signal Processor Data Manual* (SPRS708) for more information.

2 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all DSP devices and support tools. Each DSP commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g., TMS320CC6672). Texas Instruments recommends one of two possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS).

Device development evolutionary flow:

TMX	Experimental device that is not necessarily representative of the final device's electrical specifications
TMP	Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification
TMS	Fully-qualified production device

Support tool development evolutionary flow:

TMDX	Development-support product that has not yet completed Texas Instruments internal qualification testing
TMDS	Fully-qualified development-support product

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

Developmental product is intended for internal evaluation purposes.

TMS devices and TMDS 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 prototype devices (TMX or TMP) 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.

3 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 CYP package is shown in Figure 1. Figure 1 also shows an example of C6672 package symbolization.

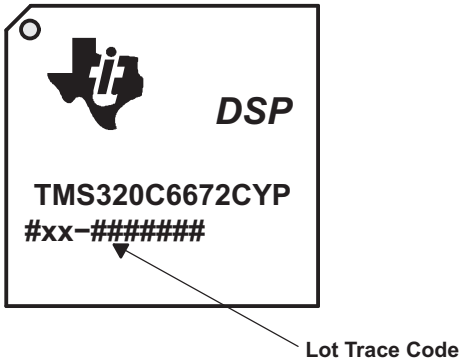


Figure 1. Lot Trace Code Example for TMS320C6672 (CYP Package)

Silicon revision correlates to the lot trace code marked on the package. This code is of the format #xx-#######. 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 TMS320C6672 devices.

Table 1. Lot Trace Codes

Lot Trace Code (xx)	Silicon Revision	Comments
10	1.0	Initial silicon revision
20	2.0	Silicon revision 2.0

The TMS320C6672 device contains multiple read-only register fields that report revision values. The JTAG ID (JTAGID), C66x CorePac Revision ID (MM_REVID) and CPU Control Status (CSR) registers allow the customer to read the current device and CPU level revision of the TMS320C6672.

The JTAG ID register (JTAGID) is a read-only register that identifies to the customer the JTAG/Device ID. The value in the VARIANT field of the JTAG ID Register changes based on the revision of the silicon being used.

The C66x CorePac Revision ID register (MM_REVID) 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 *TMS320*C6672 *Multicore Fixed and Floating-Point Digital Signal Processor Data Manual* (SPRS708).

The CPU Control Status Register (CSR) contains a read-only REVISION_ID field that identifies to the customer the revision of the CPU being used. More information about the CPU Control Status Register can be found in the *C66x CPU and Instruction Set Reference Guide* (SPRUGH7).

Table 2 shows the contents of the CPU Control Status Register CPU_ID and REVISION_ID fields, C66x CorePac MM_REVID Register REVISION field, and the JTAGID register VARIANT field for each silicon revision of the C6672 device.

Table 2. Silicon Revision Variables

Silicon Revision	CPU CSR Register	C66x CorePac MM_REVID Register	C6672 JTAGID Register
1.0	CSR[CPU_ID] = 15h	Rev. 1.0	JTAGID = 0009E02Fh
	CSR[REVISION_ID] = 00h	MM_REVID[REVISION] = 0000h	
2.0	CSR[CPU_ID] = 15h	Rev. 2.0	JTAGID = 1009E02Fh
	CSR[REVISION_ID] = 00h	MM_REVID[REVISION] = 0001h	

More details on the JTAG ID and CorePac Revision ID Registers can be found in the *TMS320*C6672 *Multicore Fixed and Floating-Point Digital Signal Processor Data Manual* (SPRS708).

4 Silicon Updates

Table 3 lists the silicon updates applicable to each silicon revision. For details on each advisory, click on the link below.

Table 3. Silicon Revision 1.0 and 2.0 Updates

Silicon Update Advisory	See	Applies To Silicon Revision	
		1.0	2.0
SRIO Deadlock Issue	Advisory 1	X	
SRIO Potential Message Header Corruption Issue	Advisory 2	X	
HyperLink Temporary Blocking Issue	Advisory 3	X	X
SRIO Reset Isolation Not Functional Issue	Advisory 4	X	
SRIO Type 9 / Type 11 RX Message Uses Wrong Context Issue	Advisory 5	X	
Total Number of SRIO RXU Contexts Decreases Due to Timeout or Teardown Issue	Advisory 6	X	
Potential L2 Cache Corruption During Block Coherence Operations Issue	Advisory 7	X	
Multiple PLLs May Not Lock After Power-on Reset Issue	Advisory 8	X	
DDR3 Automatic Leveling Issue	Advisory 9	X	X
Queue Manager External Linking RAM Location Issue	Advisory 10	X	X
DDR3 Excessive Refresh Issue	Advisory 11	X	X
SRIO Packet Forwarding Issue	Advisory 12	X	
SRIO Messaging in Highly Oversubscribed System Issue	Advisory 13	X	
SRIO Direct I/O NREAD Data Corruption Issue	Advisory 14	X	
SRIO Register Aliasing Issue	Advisory 15	X	
SRIO Status Control Symbols Are Sent More Often Than Required Issue	Advisory 16	X	X
Corruption of Control Characters in SRIO Line Loopback Mode Issue	Advisory 17	X	X
SerDes Transit Signals Pass ESD-CDM up to +/-150V Issue	Advisory 18	X	X
$\overline{\text{RESETSTAT}}$ Signal Driven High Issue	Advisory 19	X	X
Corruption of Read Access to PCIe Registers in Big Endian Mode Issue	Advisory 20	X	
HyperLink Data Rate Limited to 40Gbaud Issue	Advisory 21	X	X
L2 Cache Corruption During Block and Global Coherence Operations Issue	Advisory 22		X
SerDes AC-JTAG (1149.6) Receiver Sensitivity Issue	Advisory 23	X	X
System Reset Operation Disconnects SoC from CCS Issue	Advisory 24		X
Power Domains Hang on Power-Up at the Same Time as a $\overline{\text{RESET}}$ (Hard Reset) Received Issue	Advisory 25	X	X
DDR3 Incremental Write Leveling Issue	Advisory 26	X	X
Single MFENCE Issue	Advisory 27	X	X
Read Exception and Data Corruption Issue	Advisory 28	X	X
False DDR3 Write ECC Error Reported Under Certain Conditions	Advisory 29	X	X
Descriptors Placed in PCIe Memory Space can Cause Problems	Advisory 30	X	X

Table 3. Silicon Revision 1.0 and 2.0 Updates (continued)

Silicon Update Advisory	See	Applies To Silicon Revision	
		1.0	2.0
PCIe Device ID Field Reset Value Usage Note	Usage Note 1	X	
Packet DMA Does Not Update RX PS Region Location Bit Usage Note	Usage Note 2	X	X
Packet DMA Clock-Gating Usage Note	Usage Note 3	X	X
Disable KICK Registers Usage Note	Usage Note 4	X	X
DDR3 ZQ Calibration Usage Note	Usage Note 5	X	X
I^2C Bus Hang After Master Reset Usage Note	Usage Note 6	X	X
$\overline{POR}$ and $\overline{RESETFULL}$ Sequence Usage Note	Usage Note 7	X	X
MPU Read Permissions for Queue Manager Subsystem Usage Note	Usage Note 8	X	X
PLL ENSAT Bit Usage Note	Usage Note 9	X	
Queue Proxy Access Usage Note	Usage Note 10	X	X
Minimizing Main PLL Jitter Usage Note	Usage Note 11	X	X
SRIO and PA_SS PKTDMA RX Descriptor Buffer Size Usage Note	Usage Note 12	X	
PLL Boot Configuration Settings and DEVSPEED Register Usage Note	Usage Note 13	X	
DDR3 Performance Limited to 1333MT/s Usage Note	Usage Note 14	X	X
PCIe BAR5 Window Size Configuration in Boot ROM Usage Note	Usage Note 15	X	
Sticky Bits in PCIe MMRs Usage Note	Usage Note 16	X	X
The Clock Input to NETCP Usage Note	Usage Note 17	X	X
Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note	Usage Note 18	X	X
CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note	Usage Note 19	X	X
IDMA1 Performance Limitation Usage Note	Usage Note 20	X	
Revised PLL Programming Sequence Usage Note	Usage Note 21	X	X
Core Wake Up on $\overline{RESET}$ Usage Note	Usage Note 22		X
BSDL Testing Support Usage Note	Usage Note 23	X	X
Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note	Usage Note 24	X	X
Performance Degradation for Asynchronous Accesses Caused by An Unused Feature Enabled in EMIF 16 Usage Note	Usage Note 25	X	X
DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note	Usage Note 26	X	X

Silicon Updates

Title **Page**

Advisory 1 *SRIO Deadlock Issue*

Revision(s) Affected: 1.0

Details: At least two devices exchanging a continuous stream for some period of time of SRIO transactions-with-responses may encounter a deadlock. SRIO transactions with responses include NREAD, NWRITE_R, DOORBELL, and Type 11 messages. SRIO transactions without responses are not impacted. SRIO transactions without responses include NWRITE, SWRITE, and Type 9 packets.

Deadlock results because both devices have requests being processed in the receive part (MAU and RXU) that must provide responses. The inbound buffers in the logical layer are getting filled with incoming requests and responses. The responses in the inbound buffers are blocked by the requests that are being processed. The outbound buffers in the logic layer also become filled with requests and responses making it impossible for the MAU and RXU to obtain the buffer credit that is needed to provide the response and get to the next request or response in the inbound buffer.

Workaround 1: Use mostly transactions that do not require responses. If responses are infrequent then transactions will continue to progress until the response can be returned. Use Type 9 packets instead of Type 11 messages. Use NWRITE (push) rather than NREAD (pull) and in preference to NWRITE_R. An occasional NWRITE_R to confirm that data has been successfully written or DOORBELL to alert the other device is acceptable.

Workaround 2: Limit Type 11 messages and DIO blocks to one (or possibly two) SRIO segments. This limits the number of buffers (segments) that can be outstanding. Impacts of this method are as below:

- For Type 11 messages this increases overhead with respect to the Queue Manager and the management of the buffer descriptors.
- For DIO (NREAD, NWRITE_R) this increases overhead with respect to programming the LSUs.

Step 1 to Minimize Risk of Issue: The 2× mode of operation has a higher probability of deadlock occurrence than the 1× mode because the 2× mode has more physical layer outbound buffers than the 1× mode, which can increase the egress traffic, which, in turn, can fill the ingress buffers of the link partner with low priority traffic. So, changing from 2× mode to 1× mode will reduce the probability of occurrence of this deadlock. Additional reduction of the egress packet rate can be done by using priority 0 only for outbound requests, which has the least number of outbound buffers, therefore reducing the traffic. Further reduction of the egress rate could be managed with outbound watermark settings, which can reduce the number of required outbound buffers for a particular priority.

Step 2 to Minimize Risk of Issue: For DIO transactions, use only a single LSU between two devices, which limits the max outstanding non-posted packets to 16. The LSU will naturally progress. Outbound buffer usage will depend on the responses. Impacts of this method are as below:

- This allows hardware to throttle the non-posted requests to 16 at any given time.
- However, the LSUs have shadow registers providing a means of programming a transaction while another transaction is in progress. The maximum for a single LSU is 9, each supporting 1-MB transactions.
- Some head of line blocking (HOL) characteristics may arise due to larger blocks (presumably for a lower priority transaction) having to complete before the next transaction can proceed.
- Increased overhead to manage sharing of a single LSU among the cores especially if block sizes are reduced to avoid the possible HOL characteristics.

Step 3 to Minimize Risk of Issue: Use a single priority for all transactions, and allow for auto priority promotion to make it easier for responses to get through the transmit path. Impact is that this may cause some HOL characteristics, especially for Type 11 messages.

Step 4 to Minimize Risk of Issue: Limit LSU/TXU channels to one per destination. Because, at the same priority, this will guarantee round robin, which will effectively interleave outgoing requests and spread out requests to an individual destination.

Advisory 2 *SRIO Potential Message Header Corruption Issue*

Revision(s) Affected: 1.0

Details: This issue can occur if both SRIO Type 9 and Type 11 messaging are used in the same device. When the RXU submodule inside the SRIO module times out while waiting for the rest of the data of a multi-segment message, an incorrect message type may be used to construct the protocol-specific (PS) data in the returned message header. The PS data is filled according to the last message type received instead of the message type that generated the timeout. Because Type 9 and Type 11 have different structures, if the wrong type is used, incorrect fields are used to fill the PS data in the RX packet.

Workaround 1: The DSP can detect the error condition using certain bits in the received PS data. This requires the knowledge of the SRIO TT setting used in the system. It also requires always setting the least-significant bit of the StreamID to 0, i.e. using only even StreamIDs. The following algorithm can then be used.

```
if (Pkt_type in Navigator descriptor == 31) (Type 11 RX packet was received)
{

 if (known value of TT used in system == 0)
 {

  if (PS info word 2, bit 9 == ‘1’)
   Header corruption occurred. RX packet actually timed out.
  else
   Header corruption did NOT occur. Process RX packet normally.
 }
 else (known value of TT used in system == 1)
 {
  if (PS info word 2, bit 10 == ‘1’)
   Header corruption occurred. RX packet actually timed out.
  else
   Header corruption did NOT occur. Process RX packet normally.
 }
}
else if (Pkt_type in Navigator descriptor == 30) (Type 9 RX packet was received)
{
 if (PS info word 2, bit 16 == ‘1’)
  Header corruption occurred. RX packet actually timed out.
 else
  Header corruption did NOT occur. Process RX packet normally.
}
```

Workaround 2: Avoid mixing Type 9 and Type 11 messages.

Advisory 3 *HyperLink Temporary Blocking Issue*

Revision(s) Affected: 1.0, 2.0

Details: A temporary blocking condition can occur on HyperLink on a Keystone I local (slave) device while returning high read activity initiated by a remote device (master.) If the read accesses are sufficiently high from the remote devices, the responses from the local device can keep the response path continuously busy. As a result, the local device that is returning the read data cannot issue any commands to the remote device since the read return data is highest priority. This is called arbitration head of line blocking (HOLB). Usually to initiate such high read activity, it is expected that the remote device is using master peripheral like EDMA issuing the reads, where the read burst size is 64 byte or higher, there needs to be multiple outstanding read requests and the read accesses are initiated to high speed memory like L2 or MSMC on the local device. Typically read accesses initiated to external memory (DDR3) will not see this issue, as the read responses are slower relative to on chip memory, additionally if the read accesses are initiated by CPU, it is not expected to cause HOLB, as the CPU waits for read response data prior to issuing subsequent read commands, which limits both the read activity and number of outstanding transactions.

Workaround 1: Use a push messaging model instead of pull, if possible.

Workaround 2: If a pull model is required, use CPU for reading instead of EDMA for local (high speed/low latency) memories, so the schedule breaks when the return path is continuously busy and HOLB is avoided.

Advisory 4 *SRIO Reset Isolation Not Functional Issue*

Revision(s) Affected: 1.0

Details: The SRIO module has the capability via reset isolation to continue forwarding SRIO packets intended for other devices while the rest of the local device containing the SRIO module is in reset. However, the SRIO reset isolation feature is not supported on the device silicon revision(s) listed above. Instead of using the reset isolation feature, the software should configure the SRIO to be reset every time a device reset is asserted to the device.

Workaround: None.

Please note that the BootROM sets up the SRIO to use reset isolation by default, but it can be disabled by software after boot up. To disable the SRIO reset isolation feature, the RESETISO bit (bit 12) in the PSC MDCTL11 register (at address 02350A2Ch) must be set to ‘0’. Also, bit 9 in the PLL controller RSISO register (at address 023100F0h) must be set to ‘0’. A read-modify-write should be used in both cases to make sure the rest of the bits in the registers are not affected.

Advisory 5 *SRIO Type 9 / Type 11 RX Message Uses Wrong Context Issue*

Revision(s) Affected: 1.0

Details: This issue can occur when a Type 9 or Type 11 SRIO multi-segment message has been partially received by the device, meaning that at least the first segment has been received, but the last segment has not yet arrived. If a second message of the opposite type is received during this time with the same header identification information as the first, the second message will incorrectly use the SRIO receive context assigned to the first message. This happens because the hardware logic does not check the message type before selecting the context based on the message identification information.

Workaround: In order for this issue to occur, the bits making up the message identification information must match between the two packets arriving at the same time. The issue is avoided if the information for packets arriving at the same time is different. One way to accomplish this would be to make sure that the least-significant six bits of the COS field for any Type 9 messages do not match the 6-bit mailbox field for any Type 11 messages that have the same src_id and dest_id.

Advisory 6 *Total Number of SRIO RXU Contexts Decreases Due to Timeout or Teardown Issue*

Revision(s) Affected: 1.0

Details: The total number of SRIO RXU contexts that can be used to store received messages may decrease when an RXU timeout or teardown occurs. Eventually, if timeouts and teardowns keep occurring, there may be no contexts available and no messages can be received.

Workaround: The RXU teardown will not cause the issue to occur if the RXU is reset (by disabling and then enabling the submodule) after the teardown is performed.

To Minimize Risk of Issue: The possibility of an RXU timeout can be minimized by making sure the device is not over-subscribed.

Advisory 7 — *Potential L2 Cache Corruption During Block Coherence Operations Issue*

Revision(s) Affected: 1.0

Details:

A potential L2 cache corruption issue during block coherence operations has been identified. Under a specific set of circumstances, L1D or L2 block coherence operations can cause L2 cache corruption. The problem arises when the following four actions happen back to back in the same L2 set:

1. L1D write miss
2. Victim writeback due to block coherence operations
3. Write allocate for some address
4. Read or write allocate for some address

This issue applies to all the block coherence operations listed below:

- L1D writeback
- L1D invalidate
- L1D writeback with invalidate
- L2 writeback
- L2 invalidate
- L2 writeback with invalidate

Workaround:

The workaround requires that the memory system be idle during the block coherence operations. Hence programs must wait for block coherence operations to complete before continuing. This applies to L1D and L2 memory block coherence operations. To issue a block coherence operation follow the sequence below.

1. Disable interrupts.
2. Write the starting address to the corresponding BAR register.
3. Write the word count to the corresponding WC register.
4. Wait for completion by one of the following methods:
   (a) Issue an MFENCE instruction (preferred).
   (b) Poll the WC register until the word count field reads as 0.
5. Perform 16 NOPs.
6. Restore interrupts.

For further information about the cache control registers (BAR and WC) see the *TMS320C66x DSP CorePac User Guide* (SPRUGW0). The MFENCE instruction is new to the C66x DSP. It stalls the DSP until all outstanding memory operations complete. For further information about the MFENCE instruction, see the *C66x DSP and Instruction Set Reference Guide* (SPRUGH7).

Advisory 8 *Multiple PLLs May Not Lock After Power-on Reset Issue*

Revision(s) Affected: 1.0

Details:

After power-on reset release, the boot mode for the device is selected based on the status of the BOOTMODE pins. The main PLL stays in bypass mode for no-boot, SPI and I^2C boot. For all other boot modes, a PLL initialization sequence executes inside the boot ROM to configure the main PLL in PLL mode.

In order to ensure proper PLL startup, the PLL power_down pin needs to be toggled. This is accomplished by toggling the PLLPWRDN bit in the PLLCTL register. This needs to be done before the main PLL initialization sequence. The PLL initialization sequence in the boot ROM does not toggle the PLLPWRDN bit. So it may cause the main PLL and/or DDR3 PLL and/or PA PLL not to be locked.

The common symptoms of this issue would be: not being able to connect in CCS, not being able to read or write DDR configuration registers, and/or the SYSCLKOUT pin not showing SYSCLK.

Workaround:

The effect of the incomplete PLL initialization sequence in the boot ROM can be avoided using the following steps:

1. Perform a read/modify write to bit-1 (PLLPWRDN) of the PLL control register (PLLCTL). The address of the PLL control register is 0x0231_0100.
2. Stay in a loop such that the bit is set for 5 µs (minimum) and then clear the bit.

Steps 1 and 2 must complete before the main PLL initialization sequence executes from the boot ROM. These two steps must also be executed for the DDR3 PLL and the PA PLL before they are configured. These two steps control all the above mentioned three PLLs (Main, DDR3 and PA). Below are the reserved pins on the device which will indicate the status of the main PLL lock, DDR3 PLL lock and PA PLL lock.

- RSV20 - COREPLLLOCK
- RSV21 - DDR3PLLLOCK
- RSV22 - PAPLLLOCK

If those pins are high then respective PLL is locked and if those pins are low then the respective PLL is not locked. Above pins can be checked to identify whether or not the device encountered the error condition.

The workaround could be implemented in one of several ways for different chip states:

- **For CCS emulation mode with a no-boot configuration:** The workaround is inside a GEL file provided by TI.
- **For no-boot, SPI and I^2C boot modes:** The workaround should be added before the main PLL initialization sequence inside the application code. Please see the I^2C boot file provided by TI.
- **For other boot modes:** The workaround is to force the chip to start an I^2C boot with a boot configuration table. The boot configuration table is used to poke a small program (the fix in steps 1 and 2) to execute before the main PLL initialization code in the device boot ROM runs. An example for this implementation is provided in above mentioned I^2C boot file. In this scenario, the I^2C primary boot implements the workaround and then configures the appropriate peripheral for a secondary boot.

Advisory 9 *DDR3 Automatic Leveling Issue*

Revision(s) Affected: 1.0, 2.0

Details:

The DDR3 PHY integrated into the DDR3 Memory Controller of the device has a problem with one of the read-write leveling hardware features.

The read-write leveling hardware features are described in Read-Write Leveling section of the *KeyStone Architecture DDR3 Memory Controller User Guide* (SPRUGV8). The three leveling types are: Write Leveling, Read DQS Gate Training and Read Data Eye Training.

The leveling feature with the problem is the Read Data Eye Training. Depending on the jitter seen on the DQ and DQS signals, this leveling feature will fail to converge on a good data eye, resulting in corrupted DSP to DDR3 SDRAM reads.

The intended purpose of the Read Data Eye Training feature is to align the DQS sampling transitions in the middle of the DQ data eye within individual byte lanes of the DDR3 memory interface. As seen in Figure 2, the DQS signal can be delayed relative to the DQ signal within the DDR3 PHY.

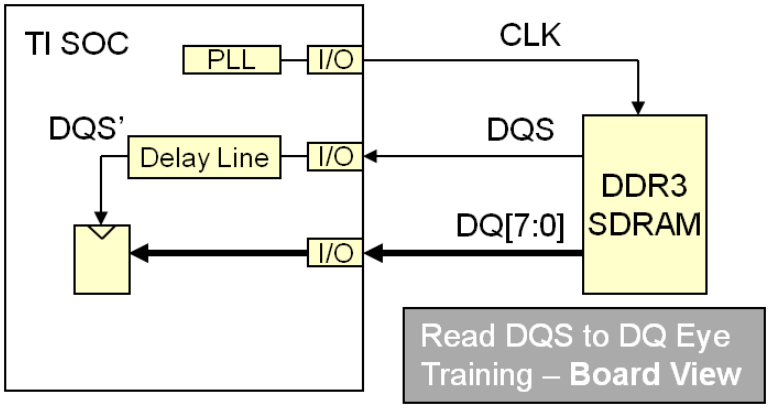


Figure 2. Read DQS to DQ Eye Training - Board View

As shown in Figure 3, the Read Data Eye Training algorithm should correctly detect the extents of the DQ data eye. The SDRAM are put into a DQS to DQ eye training mode and read back an alternating 0, 1 pattern.

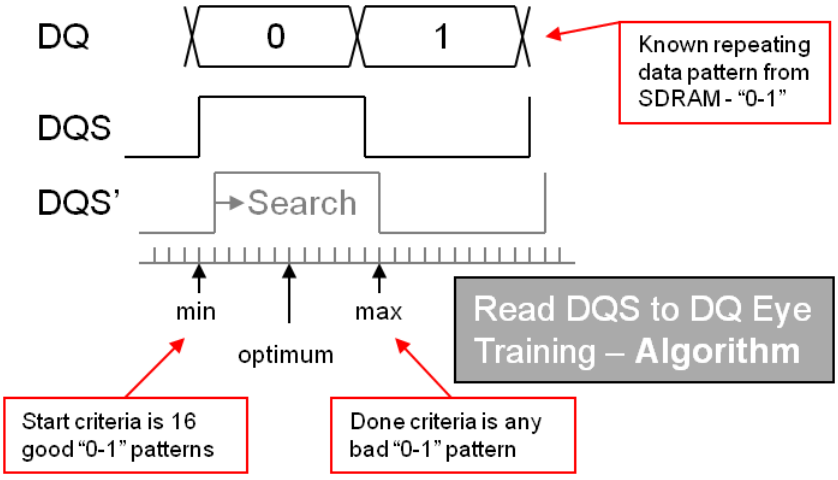


Figure 3. Read DQS to DQ Eye Training - Algorithm

The DDR3 PHY uses this pattern to detect good or bad data being sampled by the DQS transitions. The DDR3 PHY searches by delaying DQS relative to DQ until these constraints are found.

However, if too much jitter is present on the DQ or DQS signals, the search can converge to a false edge as shown in Figure 4.

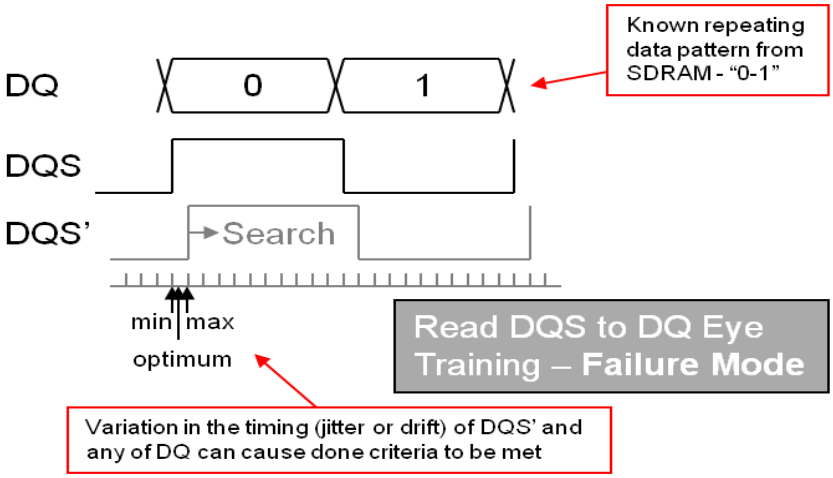


Figure 4. Read DQS to DQ Eye Training - Failure Mode

Workaround 1:

The DDR3 PHY in the device has the ability to complete auto-leveling of the write leveling values and the read DQS gate training values separate from the read data eye training. Then a fixed value is used as the read data eye sample point. This solution is functional on standard DDR3 fly-by layouts and is referred to as Partial Automatic Leveling. It has been validated for robust operation at DDR3-1333 when connected to either a UDIMM or with a discrete SDRAM implementation.

This mode is enabled by setting bit 9 (0-indexed) of the DDR3_CONFIG_REG_23 at address 0x02620460. The lower 8 bits (bits 7:0) must not be overwritten and will contain their default value of 0x34 . The default value is recommended for use. After programming DDR3_CONFIG_REG_23, proceed to enable auto-leveling. Please refer to the *DDR3 Initialization Application Report* (SPRABL2) for details of the sequence of steps.

Workaround 2:

The user has the ability to completely disable the automatic leveling features of the DDR3 controller and rely exclusively on a set of ratio-forced register values. These values control the DQ and DQS delay between all byte lanes for gate leveling, write leveling, and read data eye training. The specific registers and values to set in the registers are described in the *KeyStone Architecture DDR3 Memory Controller User Guide* (SPRUGV8). The values programmed are dependent on the specific board characteristics. Limited support is available for this solution. The customer will need to assume responsibility for validating all required timing margins are met.

Because these delay values cannot be tailored to specific byte lanes, only layouts with very small signal skews between DRAMs can use this workaround. Rather than following the routing guidelines which allow different lengths for each byte lane, all signals must be closely matched. Specifically:

1. Route lengths for address, command, control and clock between the DSP and DRAMs must be very similar. We understand that the fly-by topology prevents these lengths from matching.
2. All DQ, DQM and DQS signals must be skew-matched to within ± 10.00 mils across all byte lanes. This length should be the average of the fly-by lengths measured from the DSP to the DRAMs.

These additional routing skew constraints attempt to minimize the delay variance as much as possible, which enables the single set of values to adequately control the PHY timing. The time variation from the first to the last DRAM reduces the available sampling window. This restriction means that use of this work-around reduces the DDR3 operating speed and it can only be used with topologies with a small number of devices.

It is estimated that the following rates can be achieved based on the routing skew on the board when using ratio-forced register settings:

Table 4. Routing Skew vs. Maximum Data Rate

Maximum Skew	Maximum Data Rate
20 mils	1333 MT/s
1.0 inches	1066 MT/s
2.5 inches	800 MT/s

Workaround 3: This workaround is available only on revision 2.0 of the device. The read data eye sample point can now be optimized by incremental read eye leveling. To do this, leave the read data eye training enabled (leave bit 9 in DDR3_CONFIG_REG_23 = 0), trigger automatic leveling and then follow up with at least 64 read data eye incremental leveling events. The incremental leveling events converge the read data eye sample point to a robust sample location. This solution is functional on standard DDR3 fly-by layouts and is referred to as Full Automatic Leveling.

The time between incremental leveling events is set by the incremental leveling prescalar and incremental data eye training interval fields in the RDWR_LVL_CTRL register at 0x210000DC. The initialization routine can enable the incremental read eye leveling, pause long enough to allow the 64 events to occur and then it can disable the incremental read eye leveling, if desired. When implementing this workaround, the DDR3 interface must not be used until after these 64 incremental read eye leveling events are completed.

Advisory 10 *Queue Manager External Linking RAM Location Issue*

Revision(s) Affected: 1.0, 2.0

Details: When the Queue Manager is setup to use an external linking RAM and the external linking RAM is placed in the same memory endpoint as the packet descriptors or data buffers, then the Queue Manager may encounter a stall from which it cannot recover. The different memory endpoints are L2RAM, MSMC SRAM, and DDR3.

Workaround: The Queue Manager's external linking RAM must be placed in a different memory endpoint from the packet descriptors or data buffers. As an example, if the external linking RAM is placed in MSMC SRAM, then the packet descriptors and data buffers must be placed in L2RAM and/or DDR3. Note that the packet descriptors and data buffers do not have to be placed in the same memory endpoints.

Advisory 11 *DDR3 Excessive Refresh Issue*

Revision(s) Affected: 1.0, 2.0

Details: DDR3 JEDEC standard specifies that at any given time, a maximum of 16 refresh commands can be issued within a 2 × tREFI interval (2 × 7.8 = 15.6 µs). Failing to meet this requirement could result in a high current draw and the possibility of DDR3 device failure. The DDR3 controller will violate the above requirement if the following actions occur:

1. The DDR3 memory is put into self-refresh mode by setting the LP_MODE field in the Power Management Control register to 0x2.
2. One or more read/write commands are sent by the DDR3 controller while the DDR3 memory is in self-refresh such that the memory exits self-refresh to execute commands.
3. After command execution is complete and there are no more commands to execute, the DDR3 controller is then idle for a certain number of DDR clock cycles before putting the external memory in self-refresh mode. The number of idle DDR clock cycles is defined by SR_TIM value field in the Power Management Control register.
4. Because the DDR3 controller issues one refresh command on self-refresh entry and another refresh command on self-refresh exit, if this sequence repeats more than eight times within a 2 × tREFI interval, the DDR3 controller will issue more than 16 refresh commands in a 2 × tREFI interval and violate the JEDEC requirement.

Note if SR_TIM value is greater than or equal to 0x9, the DDR3 controller does not violate the JEDEC requirement. This is because the DDR3 controller will wait for at least 4096 clock cycles of idle time before putting DDR3 memory into self-refresh mode. Therefore, the only possible way the above scenario could occur is by setting the LP_MODE field to 0x2 to put the DDR3 memory in self-refresh mode with the SR_TIM value field less than 0x9, and then sending periodic read/write commands.

Workaround: When using LP_MODE=0x2 to enter self-refresh mode, SR_TIM needs to be programmed greater than or equal to 0x9. For further information about the Power Management Control register see the *KeyStone Architecture DDR3 Memory Controller User Guide* (SPRUGV8).

Advisory 12 *SRIO Packet Forwarding Issue*

Revision(s) Affected: 1.0

Details: When a high volume of SRIO packets with varying priorities arrive at a device and are being forwarded to another device, there is a possibility that the local SRIO module will lock up and stop processing incoming packets. The logic in the SRIO module places all packets to be forwarded in the same internal queue, allowing the possibility that a high volume of lower priority packets may block a higher priority packet and cause all packet processing to stop.

Workaround: Pace the packet forwarded traffic so that congestion is avoided and make all packet forwarded traffic the same priority. This priority should be higher than the other incoming traffic destined for the local device.

Advisory 13 *SRIO Messaging in Highly Oversubscribed System Issue*

Revision(s) Affected: 1.0

Details: When the SRIO message-passing mode is used in a highly oversubscribed system, it is possible for the number of available TX contexts to decrease over time and eventually cause the module to stop transmitting TX packets. An oversubscribed system is one where the number of RXU contexts required in any of the devices tends to exceed the maximum number of RXU contexts in that device. There are 16 RXU contexts per device.

There are two scenarios that can cause the failure. In the first scenario, a retry that happens at nearly the same time as a DONE response can trigger an error in the internal logic that causes a TX context to become unusable. In the second scenario, an error/timeout condition can cause a TX context to be incorrectly cleared twice, making it unusable. In either case, all TX contexts may eventually become unavailable and cause the SRIO module to stop transmitting TX packets.

To reduce the possibility of the system being oversubscribed, there should be at least one RXU context in a receiving device for every TXU in the other devices that may send a message to the receiving device. If any of the TXUs are using Type 11 messages, they should be allocated 1.25 RXU contexts in the receiving device due to possible delays in the message responses that may cause slightly more RXU contexts to be used in the receiving device.

Workaround: None.

Advisory 14 *SRIO Direct I/O NREAD Data Corruption Issue*

Revision(s) Affected: 1.0

Details: When SRIO direct I/O traffic in a system generates error conditions such as timeouts or error responses, it is possible that an error in the internal logic of the SRIO module will cause NREAD commands to return their read data to an incorrect address, thus potentially causing data corruption in the system. The issue occurs only when multiple LSUs are in use.

Workaround 1: Limit the size of all NREAD transactions to 256 bytes so they fit in a single SRIO message. This will avoid the internal error in the SRIO module that causes the issue.

Workaround 2: Use only one LSU within the system. This will prevent the corruption of LSU data used to return the NREAD payload with the correct address.

Advisory 15 *SRIO Register Aliasing Issue*

Revision(s) Affected: 1.0

Details: Due to an error in the address decoding logic of the SRIO module, writes to SRIO MAC registers with an address offset in the range of 1B400h to1B9FCh will cause the same data to also be written to the SRIO RXU mapping register with an address offset in the range of 00400h to 009FCh and whose address has the same lowest 12 bits. This issue can potentially make the RXU mapping registers have random values that might allow spurious messages to enter a mailbox or disallow valid messages from entering a mailbox.

Workaround: The MAC registers affected should be initialized before initializing the RXU registers to ensure that the RXU registers keep their correct values. The affected MAC registers should be written only during initialization. If they are written again later, they will corrupt the corresponding RXU registers.

Some SRIO MAC registers may need to be written after initialization for normal operation. They are listed in the following two tables along with their corresponding RXU mapping register. To allow the MAC registers to be written after initialization, the corresponding RXU mapping entries cannot be used by the application. This means that the affected RXU mapping registers cannot be used for normal operation and should be avoided by the application software. The Type 9 RXU mapping registers affected are RXU_TYPE9_MAP{45, 47, 49, 50, 52}. The Type 11 RXU mapping registers affected are RXU_MAP{01, 12, 22}.

The RXU mapping registers listed in Table 5 should be disabled during initialization by writing a value of **11b** to the **tt** field in the RXU_MAPxx_H or RXU_TYPE9_MAPxx_1 register for that RXU mapping entry. The base address of the SRIO block must be added to the register offsets given in the table to obtain the full address of each register.

Table 5. Affected SRIO MAC Registers That May Need to Be Written After Initialization

MAC Register Offset	MAC Register Name	RXU Register Offset	RXU Register Name
1B490h	TLM_SP2_STATUS	00490h	RXU_MAP12_L
1B510h	TLM_SP3_STATUS	00510h	RXU_MAP22_QID
1B924h	EM_PW_EN	00924h	RXU_TYPE9_MAP45_2
1B934h	EM_DEV_PW_EN	00934h	RXU_TYPE9_MAP47_0
1B93Ch	EM_MECS_STAT	0093Ch	RXU_TYPE9_MAP47_2
1B94Ch	EM_MECS_REQ	0094Ch	RXU_TYPE9_MAP49_0
1B960h	EM_RST_PORT_STAT	00960h	RXU_TYPE9_MAP50_2
1B970h	EM_RST_PW_EN	00970h	RXU_TYPE9_MAP52_0

The two MAC registers in Table 6 may also need to be written after initialization, but they require special handling because they ultimately write data to the corresponding RXU_MAPxx_H or RXU_TYPE9_MAPxx_1 register directly. The two RXU mapping registers listed should still be disabled during initialization like the other registers above.

Table 6. Affected SRIO MAC Registers That May Need to Be Written After Initialization and Require Special Handling

MAC Register Offset	MAC Register Name	RXU Register Offset	RXU Register Name
1B410h	TLM_SP1_STATUS	00410h	RXU_MAP01_H
1B950h	EM_MECS_PORT_STAT	00950h	RXU_TYPE9_MAP49_1

However, to ensure that they stay disabled, the two reserved bits in the MAC register that match the bit indices of the **tt** field in the corresponding RXU mapping register must be set to a value of **11b** whenever the MAC register is written. For example, bits [14-13] of the TLM_SP1_STATUS register must be written with a value of 11b because those bits match the tt field location in the RXU_MAP01_H register. Also, bits [14-13] of the EM_MECS_PORT_STAT register must be written with a value of 11b because those bits match the tt field location in the RXU_TYPE9_MAP49_1 register. The base address of the SRIO block must be added to the register offsets given in the table to obtain the full address of each register.

Advisory 16 *SRIO Control Symbols Are Sent More Often Than Required Issue*

Revision(s) Affected: 1.0, 2.0

Details: Control symbols are SRIO physical layer message elements used to manage link maintenance, packet delimiting, packet acknowledgment, error reporting, and error recovery. Control symbols are used to manage the flow of transactions in the SRIO physical interconnect. The SRIO Input-status control symbols communicate the status of the physical link and packets-in-flight between the two SRIO link partners.

The bandwidth of the SRIO link is reduced because status control symbols are sent more often than required. Worst case impact is a 2.73 percent reduction in bandwidth for a 1× port operating at 1.25 Gbaud. This impact is reduced to 0.1 percent for a 4× port operating at 5 Gbaud. More details about this impact on various lane speeds and port configurations can be found in Table 7.

Table 7. The Bandwidth Reduction of the SRIO Link

Lane Speed (Gbaud)	Percentage of Bandwidth Reduction		
	1x Port	2x Port	4x Port
1.25	2.73	1.37	0.68
2.5	1.17	0.59	0.29
3.125	0.86	0.43	0.21
5	0.39	0.20	0.10

Workaround: None.

Advisory 17 *Corruption of Control Characters In SRIO Line Loopback Mode Issue*

Revision(s) Affected: 1.0, 2.0

Details: The SRIO physical layer is configured in line loopback mode on a per-port basis, by setting the LLB_EN bit in PLM_SP(n)_IMP_SPEC_CTL register. In line-loopback mode, the data from the SerDes receiver is looped back to the SerDes transmitter. Figure 5 shows the line loopback from SerDes RX to SerDes TX:

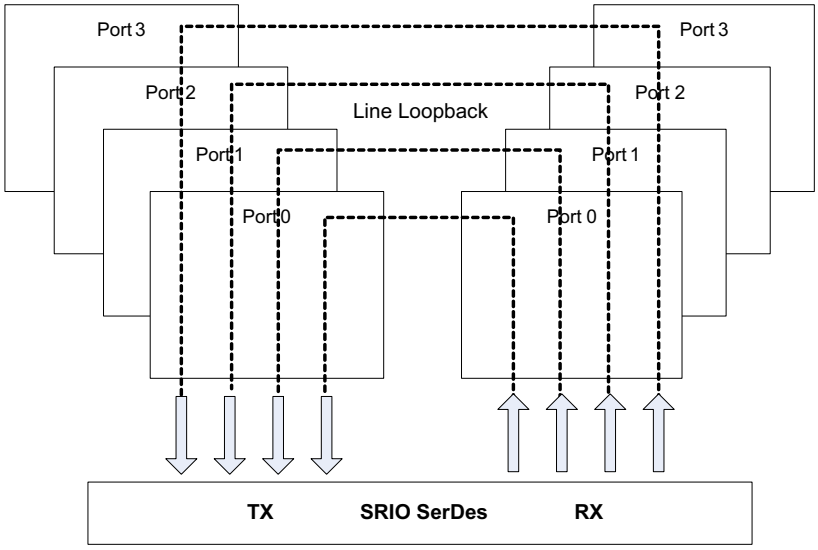


Figure 5. SRIO SerDes in Loopback Mode

The port does not provide any clock compensation when line loopback is enabled. The transmit clock must be externally synchronized with the receive clock. There is only a small FIFO that can compensate for PLL jitter or wander. Hence, correct operation of line loopback requires that the link partners use the same reference clock for the SRIO physical layer, in order to avoid overruns or underruns due to clock frequency mismatch between the link partners. As a result, line loopback mode is generally restricted to validation and qualification of board signal integrity in a lab environment.

When line loopback is enabled on one or more SRIO ports, any valid 10b code group that decodes to an illegal control character as defined by the RapidIO Specification (Revision 2.1) and whose most significant bit is 0, will be corrupted on transmission.

This issue can be summarized as follows:

- 10b code group -> Legal control character -> No problem
- 10b code group -> Illegal control character and most significant bit is 0 -> Corruption

Workaround: Instead of using PRBS sequences, users can qualify boards by using RapidIO-compliant data on the link and monitoring either per-lane error counters or port-level error counters. RapidIO compliant data is less stressful than PRBS sequences, as the RapidIO-complaint 10b data has shorter 0s and 1s run lengths than PRBS sequences. Hence, RapidIO-compliant data represent a more accurate stimulus for this test. This should be acceptable for users whose RapidIO links are of short reach, which can be either 20 cm + 1 connector or 30 cm without a connector.

Advisory 18 *SerDes Transit Signals Pass ESD-CDM up to ± 150V Issue*

Revision(s) Affected: 1.0, 2.0

Details: All data manual specifications associated with the SerDes high-speed functional pins are guaranteed to a maximum component Electrostatic Discharge - Charged Device Model (ESD-CDM) pulse threshold of 150V.

Due to the sensitive nature of the SerDes high-speed pins, exceeding component ESD CDM stress pulses of 150V on the functional pins listed below may cause permanent changes to the output swing and the de-emphasis behavior associated with these pins.

The following is the list of pins which fall into this category. Please refer to the device-specific data manual to see if these pins are present:

- Serial HyperLink Transmit Data Pins
  MCMTXN0
  MCMTXP0
  MCMTXN1
  MCMTXP1
  MCMTXN2
  MCMTXP2
  MCMTXN3
  MCMTXP3
- PCI Express Transmit Data Pins
  PCIETXN0
  PCIETXP0
  PCIETXN1
  PCIETXP1
- Serial RapidIO Transmit Data Pins
  RIOTXN0
  RIOTXP0
  RIOTXN1
  RIOTXN1
  RIOTXP1
  RIOTXN2
  RIOTXP2
  RIOTXN3
  RIOTXP3
  Ethernet MAC SGMII Transmit Data Pins
  SGMII0TXN
  SGMII0TXP
  SGMII1TXN
  SGMII1TXP

Workaround: While there is no strict workaround for this issue, there are several ways to analyze the risk with regards to CDM. Figure 6 shows the basic flow for analyzing this risk:

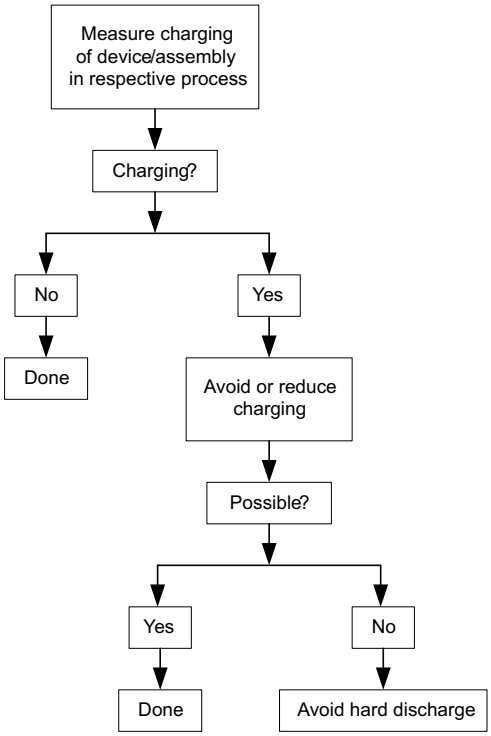


Figure 6. The Basic Flow to Analyzing the Risk

Using generally accepted *good practices* during the assembly process can help to minimize the likelihood of a hard discharge. These practices include:

- The use of an ionizer near the PCB before, during, and after placement of parts
- The use of grounded, conductive/dissipative suction cups when using pick-and-place machines
- The use of dissipative materials for downholder pins and/or plastic covers as well as two-stage pogo-pins while performing in-circuit-test

Advisory 19 — *$\overline{\text{RESETSTAT}}$ Signal Driven High Issue*

Revision(s) Affected: 1.0, 2.0

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.

Advisory 20 *Corruption of Read Access to PCIe Registers in Big Endian Mode Issue*

Revision(s) Affected: 1.0

Details: When the device operates in big endian mode, the read return data from the PCIe memory-mapped register (MMR) may be corrupted when the C66x CorePac reads the PCIe MMR value directly. Any remote read from the PCIe MMR is not impacted.

Workaround: Use EDMA to read from the PCIe MMR when the device operates in big endian mode. EDMA read access should be limited to 32 bits each time.

Advisory 21 *HyperLink Data Rate Limited to 40 Gbaud Issue*

Revision(s) Affected: 1.0, 2.0

Details: The HyperLink interface is currently limited to a maximum transfer rate of 10 Gbaud per lane (40 Gbaud for four lanes) due to a SerDes PLL limitation.

Advisory 22 *L2 Cache Corruption During Block and Global Coherence Operations Issue*

Revision(s) Affected: 2.0

Details: Under a specific set of circumstances, L1D or L2 block and global coherence operations can cause L2 cache corruption. The problem arises when the following four actions happen back-to-back in the same L2 set:

1. L1D write miss
2. Invalidate or writeback-with-invalidate due to block and global coherence operations
3. Write allocate for some address
4. Read or write allocate for some address

This issue applies to all the block and global coherence operations EXCEPT:

- L1D block writeback
- L1D global writeback
- L2 block writeback
- L2 global writeback

See also: Advisory 7 - Potential L2 Cache Corruption During Block Coherence Operations Issue

Generic Workarounds: The workarounds below are very generic and may have performance impacts. Customers are requested to understand the application and see which one suits them better.

Workaround 1: This workaround requires that the memory system be idle during the block and global coherence operations. Hence programs must wait for block and global coherence operations to complete before continuing. This applies to L1D and L2 memory block and global coherence operations.

To issue a block coherence operation, follow the sequence below:

1. Disable interrupts.
2. Write the starting address to the corresponding BAR register.
3. Write the word count to the corresponding WC register
4. Wait for completion by one of the following methods:
   (a) Issue an MFENCE instruction (preferred)
   (b) Poll the WC register until the word count field reads as 0
5. Perform 16 NOPs
6. Restore interrupts

To issue a global coherence operation, follow the sequence below:

1. Disable interrupts.
2. Write 1 to the corresponding global coherence register (L1DINV, L1DWBINV, L2DINV, and L2DWBINV)
3. Wait for completion by one of the following methods
   (a) Issue an MFENCE instruction (preferred)
   (b) Poll the corresponding global coherence register (L1DINV, L1DWBINV, L2DINV, and L2DWBINV) until the bit [0] field reads as 0
4. Perform 16 NOPs
5. Restore interrupts

Workaround 2:

This workaround is also generic, but will allow CPU traffic to go on in parallel with cache coherence operations. To issue a block coherence operation, follow the sequence below:

1. Issue a MFENCE command.
2. Freeze L1D cache
3. Start L1D WBINV
4. Restart CPU traffic
   (CPU operations happen in parallel with WBINV and do not need to wait for cache coherency operation to complete)
5. Poll the WC register until the word count field reads as 0.
6. WBINV completes when word count field reads 0
7. Issue an MFENCE command.
8. Unfreeze L1D cache.

For more information about the cache control registers (BAR, WC, L1DINV, L1DWBINV, L2DINV, and L2DWBINV) see the *TMS320C66x DSP CorePac User Guide* (SPRUGW0). The MFENCE instruction is new to the C66x DSP. It stalls the DSP until all outstanding memory operations complete. For further information about the MFENCE instruction, see the *C66x DSP and Instruction Set Reference Guide* (SPRUGH7).

Advisory 23 *SerDes AC-JTAG (1149.6) Receiver Sensitivity Issue*

Revision(s) Affected: 1.0, 2.0

Details:

In a production or laboratory test environment the IEEE 1149.1 and IEEE 1149.6 scan-chain cells are used to verify the proper connectivity of the C6672 SoC to other board components (other C6672, or third-party IC, or connectors) through the use of the JTAG BSDL scan-chain commands.

The IEEE 1149.1 scan-chain cells are designed for use with DC-coupled signals and rely on signal levels; these cells will detect normal VIH or VIL thresholds. Most I/O on C6672 devices fall under this category. The IEEE 1149.6 scan-chain cells are design for use with AC-coupled signals and rely on signal edge detection; these cells will detect falling edge transitions or rising edge transitions since DC offsets cannot be maintained through an AC-coupled signal path.

The SerDes receivers as shown in Figure 7 for AIF2, SRIO, PCIe, SGMII and Hyperlink are all designed to be AC coupled to their associated transmitters. For these AC-coupled SerDes receivers the IEEE 1149.6 scan-chain provides edge-sensitive sensing.

A problem has been identified in some cases where the AC-JTAG (IEEE 1149.6, edge-sensitive) input cells on Rincewind and Draco receivers do not reliably respond to input edge transitions from an associated transmitter. Because of this failure, the 1149.6 scan-chain users can receive faulty signal levels transitions or no transitions at all.

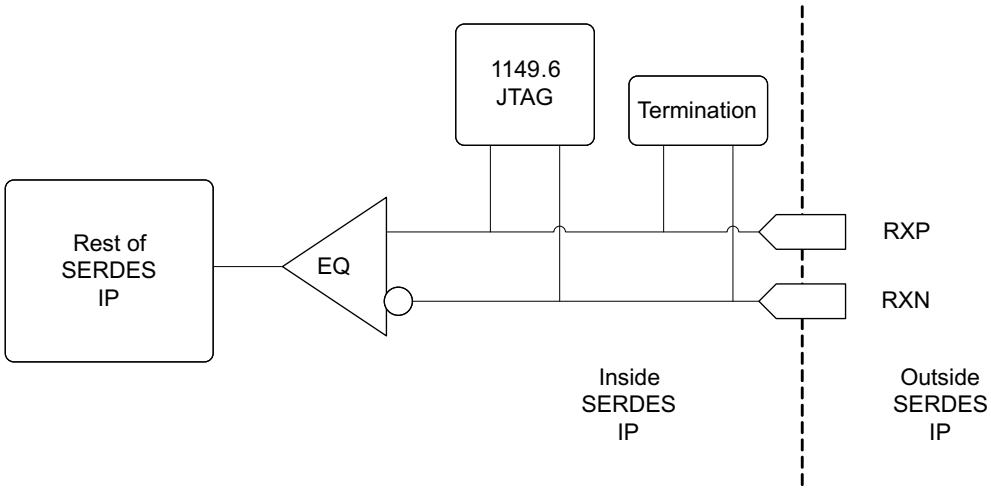


Figure 7. Simplified SerDes Receiver Block Diagram Depicting Relative Placement of Major Receiver Components

IEEE 1149.1, level-sensitive, input cells show no issue and normal operation of the SerDes receivers is also unaffected by this problem.

The reason for the issue is due to the original AC-JTAG input cell performance being overly sensitive to rise/fall times of the signals interfacing to it and input transition frequency.

The AC-JTAG input cell is based on a self-referenced, hysteretic comparator. See Figure 8. If the SerDes receiver pins are idle (no transitions are applied), the voltage of the pins will settle to the programmed, receiver common-mode voltage (0.80V nominally). From this idle state a single edge transition applied to the AC-JTAG input cell can push both terminals of the AC-JTAG comparator inputs past the VDDT (1.0V) supply voltage level.

When the comparator inputs are saturated the comparator gain is minimized and under certain PVT conditions (strong silicon, 125C junction temperature, minimum VDDT), the comparator can fail to switch to the proper state, or not respond at all.

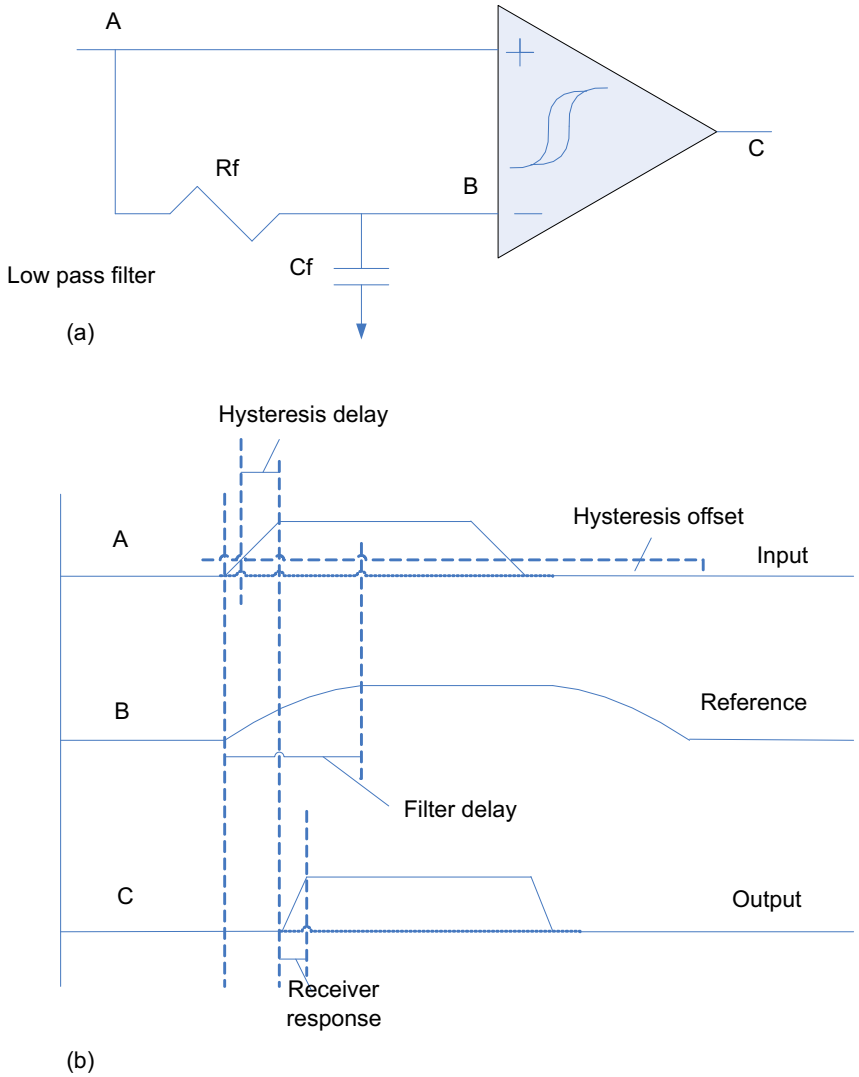


Figure 8. Self-referenced Comparator and Response

NOTE: To find the transitions in a signal, a self-referencing, hysteretic comparator (a) receives a signal, A, and a delayed version of the signal, B (b). The comparator responds to an ac-coupled signal with a reconstruction of the original (dc) signal, C.

Workaround:

One potential workaround has been identified. The failure mode of the AC-JTAG comparator is most pronounced by single edge transitions at low frequency as this creates the highest likelihood of the comparator inputs being saturated and switching incorrectly. More frequent transitions can result in proper switching of the comparator output.

The basic procedure is to initiate multiple edge transitions and then read the state of the RX input cell in question. However, due to the nature of the failure, the exact number would have to be determined experimentally for a given device.

Additionally, keeping the temperature of the device lower in the operating range and the VDDT voltage higher in the operating range can help avoid the issue.

Advisory 24 *System Reset Operation Disconnects SoC from CCS Issue*

Revision(s) Affected: 2.0

Details: The CCS connection to targets will fail after system reset is issued via CCS. The CCS connection to targets will also fail after resetting the device with the $\overline{\text{RESET}}$ pin.

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, open the CCS target configuration window and click on the sub-path of ICEPICK_D for each C66x CorePac. Then click on the property option **Domain Power Loss Mode** and select **Auto**.

Support for this new option will be released in an emupack update.

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.

Advisory 25 — *Power Domains Hang on Power-Up at the Same Time as a RESET (Hard Reset) Received Issue*

Revision(s) Affected: 1.0, 2.0

Summary: Certain power domains, like those mentioned below, have multiple RAMs and the controllers associated with them are daisy-chained. This issue occurs, when the software is powering up one of these power domains and at the same time a RESET (hard reset) is received.

Details: Power Domain affected: Power Domain 7 (MSMC RAM) and Power Domains 8-15 (CorePac 0-7, L1/L2 RAMs).

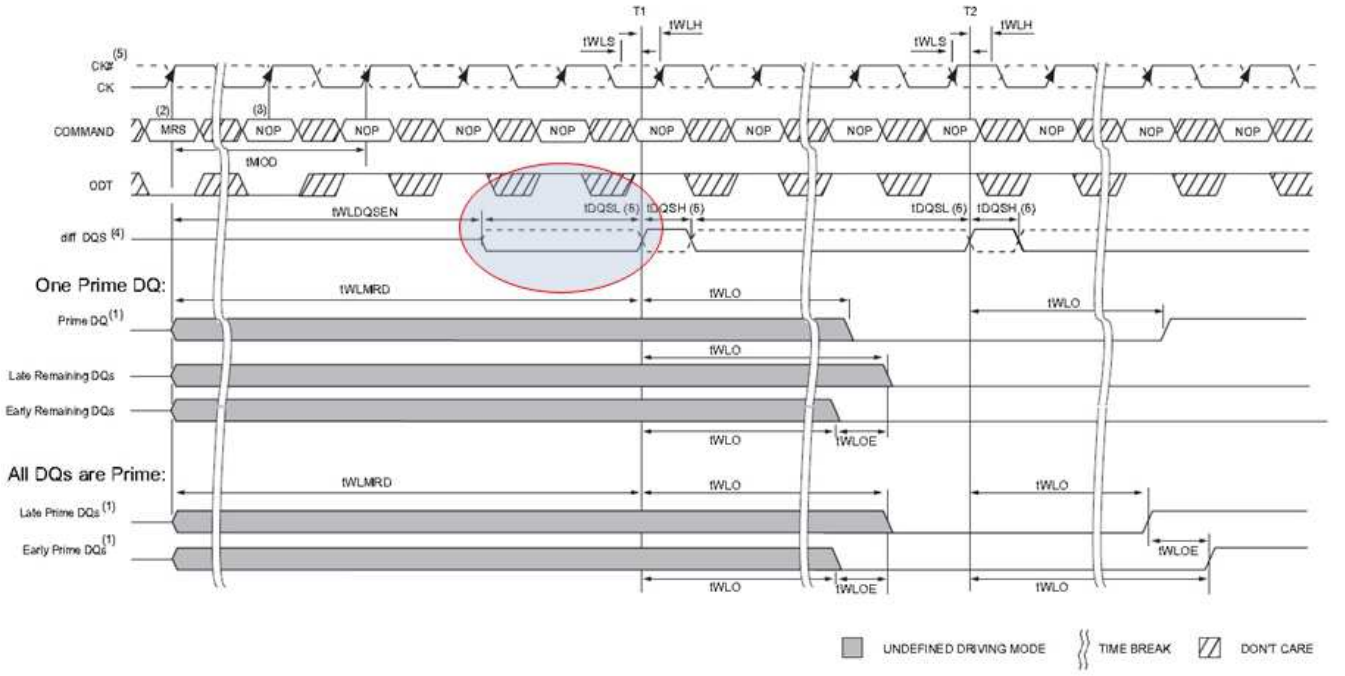
The global PSC state machine associated with the power domain in transition hangs and as a result the chip does not come out of reset as expected. The RESETSTAT pin status will be stuck low indicating that the device is in reset. The only option to exit from this hang condition is to apply a RESETFULL or POR.

Workaround: Whenever the external host controller applies a RESET (hard reset) to the device, the host is normally expected to wait for the RESETSTAT status pin to toggle from low to high (this indicates that the device is out of reset). If the RESETSTAT pin does not toggle and is stuck at low, the external host controller can infer that this issue has occurred. To recover, the external host has to apply a RESETFULL or POR to the device. Make sure that the boot configuration pins are re-latched during RESETFULL or POR.

Advisory 26 *DDR3 Incremental Write Leveling Issue*

Revision(s) Affected: 1.0, 2.0

Details: As shown in Figure 9, the DDR3 JEDEC standard requires that the DQS strobe be sent with a preamble (DQS=0) for at least tDQSL during write leveling, prior to the first rising edge of the DQS strobe (DQS=1). It was observed in simulations that the first DQS strobe is sent immediately following the high impedance state of the DQS line and without a preamble. This Z->1 transition on the DQS line may allow incremental write leveling to fail. Note that this issue impacts only the incremental write leveling feature of the DDR3 Memory controller. No issue is expected with automatic write leveling.



NOTES:
1. DRAM has the option to drive leveling feedback on a prime DQ or all DQs. If feedback is driven only on one DQ, the remaining DQs must be driven low, as shown in above Figure, and maintained at this state through out the leveling procedure.
2. MRS: Load MR1 to enter write leveling mode.
3. NOP: NOP or Deselect.
4. diff_DQS is the differential data strobe (DQS, DQS#). Timing reference points are the zero crossings. DQS is shown with solid line, DQS# is shown with dotted line.
5. CK, CK# : CK is shown with solid dark line, where as CK# is drawn with dotted line.
6. DQS, DQS# needs to fulfill minimum pulse width requirements tDQSH(min) and tDQSL(min) as defined for regular Writes; the max pulse width is system dependent.

Figure 9. Timing Details of Writing Leveling Sequence

Workaround: At this time, incremental write leveling is not supported on this device and we recommend that the incremental write leveling intervals in RDWR_LVL_CTRL and RDWR_LVL_RMP_CTRL be programmed to 0 to disable this feature.

Advisory 27 *Single MFENCE Issue*

Revision(s) Affected: 1.0, 2.0

Details:

The MFENCE instruction is used to stall the instruction fetch pipeline until the completion of all CPU-triggered memory transactions.

Under very particular circumstances, MFENCE may allow the transaction after the MFENCE to proceed before the preceding STORE completes.

For example,

1. STORE_A
2. MFENCE
3. TRANSACTION_B

The MFENCE implementation stalls the CPU until the memory system asserts that there are no transactions "in flight," i.e. it is idle. This prevents the CPU from proceeding to TRANSACTION_B before STORE_A completes. A small window exists where the memory system prematurely asserts that it is idle when STORE_A moves from L1D to L2 when it is otherwise idle. This can cause incorrect program behavior if TRANSACTION_B must occur strictly after STORE_A. For example, suppose STORE_A writes to DDR3, and TRANSACTION_B triggers an EDMA which reads the location written by STORE_A. MFENCE should guarantee that STORE_A commits before the EDMA executes, so that the EDMA sees the updated value. Due to the issue in this advisory, TRANSACTION_B could trigger the EDMA before STORE_A commits, so that the EDMA sees stale data.

Workaround:

Replace a single MFENCE with two MFENCEs back to back. This remedies the issue by resuming the stall in the case where the memory system prematurely indicated that it was idle when STORE_A passed from L1D to L2.

1. STORE_A
2. MFENCE
3. MFENCE
4. TRANSACTION_B

Note on coherence operations:

For the following advisory, double MFENCE can also be used as a workaround in addition to the workarounds already listed:

- Advisory 7 - Potential L2 Cache Corruption During Block Coherence Operations Issue
- Advisory 22 - L2 Cache Corruption During Block and Global Coherence Operations Issue

Note that the second advisory listed above does not cover L1D and L2 block and global writebacks. If L1D and L2 block and global writebacks are followed by an MFENCE and a transaction that depends on the completion of the writebacks, the double MFENCE workaround should be used.

Please also note that the current release of the MCSDK software package does not include this workaround. It will be included in a future release.

NOTE: **Special Considerations for Trace.** When trace generation is expected through software that includes the use of MFENCE, there are additional requirements for the workaround. Trace generation for MFENCE requires that every occurrence of the MFENCE instruction be followed with a NOP and a MARK instruction. The workaround is described in this white paper: http://processors.wiki.ti.com/images/c/c5/TracingMfenceWhitePaper.pdf

Advisory 28 *Read Exception and Data Corruption Issue*

Revision(s) Affected: 1.0, 2.0

Summary: Under specific circumstances, a pre-fetch for a cacheable data access (program pre-fetches are not affected) that bypasses L2 can result in a read exception and/or data corruption.

Details: A pre-fetch for a cacheable data access can result in a read exception and/or data corruption when all of the following conditions are satisfied:

1. The address being accessed lies in the range 0x0C000000 – 0xFFFFFFFF and does not lie within the CorePac's global alias 0x1n000000 – 0x1nFFFFFF, where n equals the CorePac ID number as indicated by either the DSP core number register (DNUM) or the MMID field in the L2 configuration register (L2CFG).
2. The MAR register for the given address space enables caching (MAR.PC = 1) and pre-fetch (MAR.PFX = 1).
3. L1D cache is enabled (L1DCFG.L1MODE is non-zero) and not frozen (L1DCC.OPER = 0).
4. The address does not get cached by L2 cache. This can occur for the following reasons:
   (a) The address lies in the range 0x0C000000 – 0x0FFFFFFF
   (b) The address lies above this range and L2 cache is frozen (L2CFG.L2CC = 1) or disabled (L2CFG.L2MODE = 0)

Before going into a detailed explanation of the root cause, it is worthwhile to understand the different types of XMC pre-fetch hits that can occur in a system.

1. Data Hit – An access matches an allocated entry with successful read data present in buffer. The access is serviced via the pre-fetch buffer.
2. Data Hit Wait – An access matches an allocated entry with outstanding pre-fetch reads. The access is serviced via the pre-fetch buffer when the read returns.
3. Address Hit – An access matches an allocated entry with neither successful read data nor an outstanding pre-fetch. This access is forwarded to the MSMC, but allocation for this stream will continue if applicable.
4. Miss – An access does not match an allocated entry. The access is forwarded to MSMC. If the access hits in the candidate buffer, it will be allocated as a stream.

Of the different types of pre-fetch hits listed above, this failure mode specifically requires an "Address Hit" where the allocated slot contains no data. This can happen due to a number of reasons:

1. If pre-fetches collide in the XMC pipeline, the earlier (in time) pre-fetch will be discarded.
2. When MSMC's data pre-fetch holding buffers are full, MSMC will discard the oldest pre-fetch to eliminate pre-fetch head-of-line blocking and reduce bandwidth expansion from pre-fetch.
3. The pre-fetch buffer is write-invalidate; any write that matches on an active stream invalidates any present pre-fetch data for that address.
4. Pre-fetch returned with unsuccessful read status.

The root cause of the issue is in the way the data pre-fetcher inside the XMC behaves when accessed by L1D directly (not caching in L2) as opposed to when accessed through the L2 controller (allocating in L2 cache):

The data pre-fetcher operates on 128 byte lines (the same as L2 cache line size). However, the L1D has a 64 byte line size (not matching L2 or the pre-fetch buffer).

The hardware operates differently for pre-fetches generated for L1D and L2. The L2 always consumes the entire pre-fetch line at once whereas the L1D can only consume

half of the pre-fetch line.

When an L1D access (say, to address A) hits a pre-fetch stream, the pre-fetcher may 'look back' at that stream by generating a re-fetch of the other 64 bytes of the line.

In the failing case, the hardware generates a re-fetch for an L1D access to re-fill half of the 128 byte pre-fetch line, and subsequently re-allocates that pre-fetch line to a new stream due to a subsequent data read (say, to address X). The hardware must sort the resulting pre-fetches by marking older results as stale.

The fault lies in the 'sorting' hardware which can lead to the first pre-fetch not being marked stale under certain boundary conditions (see pathological sequence below). The subsequent fetch access when the pre-fetch access wasn't marked stale will access the stale data that has been cleared (but not yet marked as stale.) This results in the read exception and/or incorrect data being fetched. Thus, the issue occurs only when the re-fetch feature is triggered and only L1D accesses use the re-fetch feature.

Since the L2 always consumes entire pre-fetch lines, it is not susceptible to the fault.

The above pre-fetch behavior leading up to the failure can be illustrated with the help of the following sequence:

Pathological sequence:

1. Initial state of the data pre-fetch buffer: Must have allocated all 8 entries [0-7] as detected streams

	Pre-fetch Buffer Initial State		
	Entry	Address	Entry Status
Allocation Pointer →	[0]	A,A+64	Stream Valid, No Data Present
	[1]	B,B+64	Stream Valid
	[2]	C,C+64	Stream Valid
	[3]	D,D+64	Stream Valid
	[4]	E,E+64	Stream Valid
	[5]	F,F+64	Stream Valid
	[6]	G,G+64	Stream Valid
	[7]	H,H+64	Stream Valid

Figure 10. Initial state of pre-fetch buffer

2. L1D pre-fetchable data read to address A hits pre-fetch entry [0] for early half of 128 byte pre-fetch line
   (a) Must be Address Hit only, both 64 byte halves of entry [0] must not contain pre-fetch data.
   (b) Entry [0] must be the 'oldest' entry, next entry to reallocate.
3. Data pre-fetcher re-fetch behavior is triggered which generates pre-fetch (PF0) to A+64.
4. PF0 stalls in XMC pipeline until step 7
5. L1D pre-fetchable data read to address X hits in candidate or pre-fetch buffer, triggering allocation of entry [0] to stream starting at X+64
6. A pre-fetch for address X+64 (PF1) is generated for entry [0] early half and followed in a subsequent cycle by X+128 (PF2).
   (a) Either PF1 or PF2 can map to the same buffer slot as PF0 (this example maps PF1)
7. PF0 transitions to the XMC output (to MSMC) on the exact same clock edge that PF1 is transitioning to the buffer allocation pipeline stage

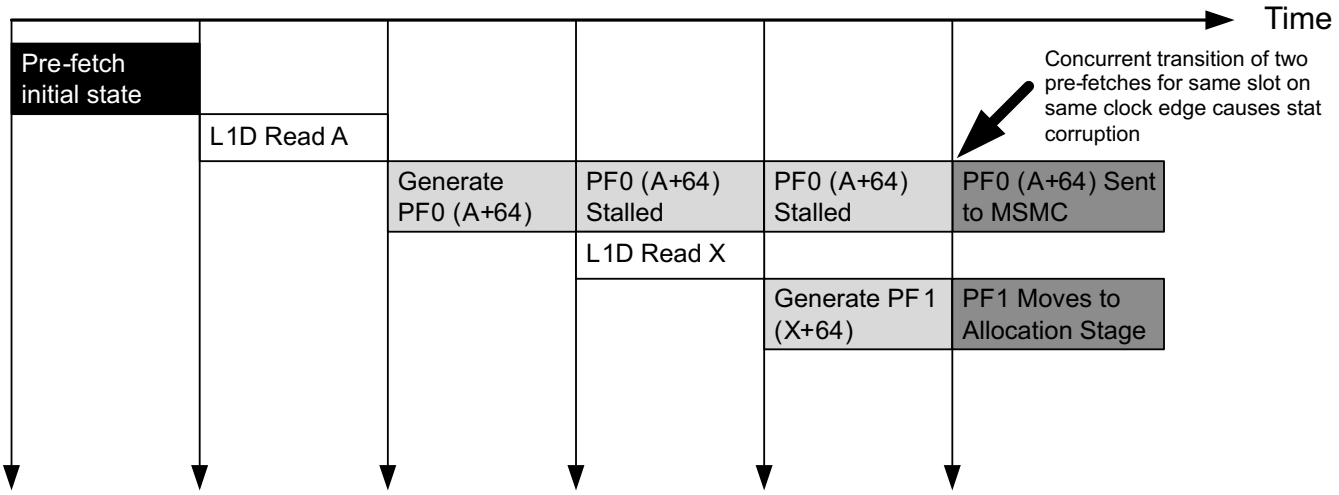


Figure 11. Sequence diagram

8. Instead of PF0 being marked stale and thrown away when returned, both PF0 and PF1 are marked valid and outstanding.
9. This leads to two valid outstanding pre-fetches for the same data buffer slot which is an invalid state and can result in unexpected behavior for a subsequent pre-fetchable data access to address X+64.

Depending on whether the subsequent pre-fetchable data access to X+64 arrives before or after the second pre-fetch returns, the result can be a read exception and/or data corruption. The table below describes the different possible outcomes depending on the relative timing between following events:

- PF0 Read Return and Status
- PF1 Read Return and Status and
- Pre-fetchable data access to X+64
  - Column 2 represents the result when the demand access arrives before the second pre-fetch return
  - Column 4 represents the result when the demand access arrives after the second pre-fetch return

Table 8. Possible Outcomes Depending on Relative Timing of Subsequent Pre-Fetchable Data Access to X+64, PF0 Return and PF1 Return

1. First Pre-Fetch Return	2. X+64 CPU Access	3.Second Pre-Fetch Return	4. X+64 CPU Access
PF0 With Data	Hit, Data Corruption	PF1 With Data	Hit, Correct
PF0 With Data	Hit, Data Corruption	PF1 Cancelled	Hit, Exception
PF0 Cancelled	**Miss, Correct**	**PF1 With Data**	**Miss, Correct**
PF0 Cancelled	**Miss, Correct**	**PF1 Cancelled**	**Miss, Correct**
PF1 With Data	Hit, Correct	PF0 With Data	Hit, Data Corruption
PF1 With Data	Hit, Correct	PF0 Cancelled	Hit, Exception
PF1 Cancelled	**Miss, Correct**	**PF0 With Data**	**Miss, Correct**
PF1 Cancelled	**Miss, Correct**	**PF0 Cancelled**	**Miss, Correct**
Color Key:	**Always Correct Operation**	Sometimes Correct Operation	Never Correct Operation

Workaround:

Software must disable the PFX bits in the MARs for address ranges 0x0C000000 – 0x0FFFFFFF corresponding to cacheable data (MARs can be written to only in supervisor mode. The PFX bit for MARs 12-15 which define attributes for 0x0C000000 – 0x0FFFFFFF is set to 1 by default). This will disable pre-fetching for accesses to those addresses, while still allowing those accesses to be cached in L1D.

If pre-fetching for MSMC SRAM and other memory spaces is desired, it can still be done provided they are remapped to a space other than 0x0C00 0000 – 0x0FFF FFFF within the MPAX registers (the remapped MSMC will act as shared level 3 memory and will be cacheable in L1D and L2).

The L2 cache must remain on and set to a cache size greater than zero, and must not be frozen when accessing pre-fetchable data, otherwise XMC will apply the previously described L1D-specific behavior for the data prefetcher and subject the system to the same issue.

Advisory 29 *False DDR3 Write ECC Error Reported Under Certain Conditions*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: 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 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 2 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 9.

Table 9. 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.
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.
10GE	sstatus, rstatus are not used	NA	NA

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

Advisory 30 *Descriptors Placed in PCIe Memory Space can Cause Problems*

Revision(s) Affected 1.0, 2.0

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.

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.

Usage Note 1 *PCIe Device ID Field Reset Value Usage Note*

Revision(s) Affected: 1.0

Details: The PCIe vendor ID / device ID field reset value is incorrect if PCIESSEN (PCIe enable/disable) pin is low at boot time (disable).

Workaround: If PCIESSEN pin is low at boot time, the vendor ID / device ID PCIe register needs to be set by software with the proper values. The vendor ID should be 0x104C. The device ID should be 0xB005. In addition, the register should be set to the correct value before enabling the link training sequence. Note: When PCIEEN is set, ROM will set the vendor ID / device ID PCIe register with the proper values. Therefore, software does not need to do it in this case.

Usage Note 2 *Packet DMA Does Not Update RX PS Region Location Bit Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: The Packet DMA inside each of the Navigator-compliant modules fails to update the Protocol-Specific Region Location bit (bit 22 of Packet Descriptor Word 0) to a 1 when it is writing an RX host-mode packet to memory with protocol-specific (PS) words located in the start of the data buffer instead of the descriptor. This means that the software cannot use this bit to determine if any PS information is located in the RX packet descriptor or at the beginning of the data buffer. The same problem will occur if the packet is sent directly to another Navigator-compliant module, as that module will not be able to determine the PS info location. This issue affects only host-type packets.

Workaround 1: Use monolithic-type packets only, thus eliminating the issue.

Workaround 2: Always place PS info in the descriptor instead of in the data buffer so that the PS location bit is always 0 and the issue does not apply.

Workaround 3: The software is responsible for configuring the Packet DMA RX flow tables, which include the PS location each flow will use. Thus, for packets sent to the DSP (not to another module directly), the software can be designed to keep track of the PS info location so it does not have to rely on the bit in the RX packet descriptor. This can be accomplished in many different ways. The following are a couple of examples:

- The software can always use the same PS location setting. This eliminates the need to find out the location from the RX descriptor.
- The software can place some form of identifier in one of the user-defined tag fields in the TX descriptor and configure the Packet DMA to pass that information through to the RX descriptor. The software can then use the identifier along with previously stored information to determine the PS info location.

Usage Note 3 *Packet DMA Clock-Gating Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: Clock-gating a module with a Navigator interface (Packet DMA) while it is writing RX packets to memory can cause undefined behavior.

Workaround: Disable/teardown all of the Packet DMA's channels before clock-gating the module in the PSC.

Usage Note 4 *Disable KICK Registers Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: The Bootcfg module contains a kicker mechanism to prevent any spurious writes from changing any of the Bootcfg register values. When the kicker mechanism is locked (which it is initially after power on reset), none of the Bootcfg MMRs are writable (they are only readable). This mechanism requires two register writes (one to KICK0 and one to KICK1) with exact data values before the kicker lock mechanism is un-locked. Once released then all the Bootcfg MMRs having write permissions are writable (the read only MMRs are still read only). The device has only one set of kicker registers to lock/unlock the Bootcfg registers. This creates potential race conditions in the multi-core environment when the different DSP cores try to access Bootcfg registers at the same time.

Workaround: The kicker mechanism is unlocked by the ROM code. Do not write any other different values afterward to these registers because that will lock the kicker mechanism and block any writes to Bootcfg registers.

Usage Note 5 *DDR3 ZQ Calibration Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: Incorrect impedance calibration will occur if DDR3 devices on the board share ZQ resistors. This is the resistor connected to the ZQ pin on a DDR3 device.

Workaround: Use independent ZQ resistors for all DDR3 devices on the board. The reg_zq_dualcalen field in the EMIF's SDRAM Output Impedance Calibration Config register must also be set to 1.

Usage Note 6 *I²C Bus Hang After Master Reset Usage Note*

Revision(s) Affected: 1.0, 2.0

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.

Usage Note 7 *POR and RESETFULL Sequence Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: For boot configuration pins to be latched correctly, during the power sequencing and reset control for chip initialization, RESETFULL must be held low for a period after the rising edge of POR but may be held low for longer periods if necessary. The configuration bits shared with the GPIO pins will be latched on the rising edge of RESETFULL and must meet the setup and hold times. Timing requirements are specified in the device-specific data manual,*TMS320*C6672 *Multicore Fixed and Floating-Point Digital Signal Processor Data Manual* (SPRS708).

Usage Note 8 *MPU Read Permissions for Queue Manager Subsystem Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: The Memory Protection Unit (MPU) has the ability to restrict the write and read permissions for bus masters like the DSP cores and System EDMAs when they attempt to access various portions of the address space on the device. One of the peripherals that may be access-controlled is the Queue Manager Subsystem (QMSS). For proper device operation, all of the read permissions for the VBUSM slave port of the QMSS must be enabled. If any of the read permissions for the VBUSM slave port of the QMSS are disabled, invalid read data and EDMA malfunction may occur. This usage note does not impact the VBUSP slave port on the QMSS or the QMSS write permissions, which may be enabled or disabled.

Usage Note 9 *PLL ENSAT Bit Usage Note*

Revision(s) Affected: 1.0

Details: For optimal PLL operation, the *ENSAT* bit in the PLL control registers for the Main PLL, DDR3 PLL, and PA PLL should be set to 1. The PLL initialization sequence in the boot ROM sets this bit to 0 and could lead to non-optimal PLL operation. Software can set the bit to the optimal value of 1 after boot. This requires setting the ENSAT bit (bit 6) to a value of 1 in the MAINPLLCTL1 register (address 0x0262032C), DDR3PLLCTL1 register (address 0x02620334), and PAPLLCTL1 register (address 0x0262033C). Read-modify-writes can be used to make sure other bits in the registers are not affected.

Usage Note 10 *Queue Proxy Access Usage Note*

Revision(s) Affected: 1.0, 2.0

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 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 the device is in 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 specific queue, C66x CorePac should use the queue peek region to get the information.

Usage Note 11 *Minimizing Main PLL Jitter Usage Note*

Revision(s) Affected: 1.0, 2.0

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 closely 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 *TMS320*C6672 *Multicore Fixed and Floating-Point Digital Signal Processor Data Manual* (SPRS708).

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

CLK = CLKIN × (PLLM+1) ÷ (2 × (PLLD+1))

Usage Note 12 *SRIO and PA_SS PKTDMA RX Descriptor Buffer Size Usage Note*

Revision(s) Affected: 1.0

Details: In the silicon revision(s) listed, the PKTDMA in the SRIO and PA_SS blocks requires the RX descriptor buffer size to be at least one byte larger than the output payload data sent in the buffer. If the RX buffer is equal in size to the output payload data, the PKTDMA will use an extra RX descriptor which it will link to the RX descriptor, even though there will be no output data placed in the extra descriptor's buffer.

Workaround: One workaround is to size the RX descriptor buffers so they are at least one byte larger than the maximum expected output payload data size.

Another workaround is to allocate enough extra RX descriptors with associated buffers to make up for the extra ones that will be required when the RX descriptor buffer is equal in size to the output payload data.

Usage Note 13 *PLL Boot Configuration Settings and DEVSPEED Register Usage Note*

Revision(s) Affected: 1.0

Details: In silicon revision 1.0, the CorePac main PLL is configured as 800MHz, 1000MHz, or 1200MHz in the BootROM code based on the DEVSPEED register reading. For the device to support a higher frequency, such as 1250MHz, the main PLL will not be configured correctly by the boot mode configurations because the DEVSPEED register reading is limited to 1200MHz.

Workaround: Reprogram the main PLL configuration settings appropriately to achieve the target device frequency at boot time of the user application. The application should follow the PLL programming procedure that is documented in the device-specific data manual.

Usage Note 14 *DDR3 Performance Limited to 1333MT/s Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: The DDR3 interface is currently limited to a maximum transfer rate of 1333MT/s. It can be configured to operate at any rate from 800MT/s to 1333MT/s. This coincides with DDR3 clock rates from 400MHz to 666.5MHz.

Usage Note 15 *PCIe BAR5 Window Size Configuration in Boot ROM Usage Note*

Revision(s) Affected: 1.0

Details: In silicon revision 1.0, the boot ROM code does not configure the PCIe BAR5 Mask Register. The BAR5 address space (window size) is disabled after boot when operating PCIe as an endpoint (EP).

Workaround: If BAR5 is required when operating PCIe as EP, the user application can configure the BAR5 Mask Register on the EP side prior to the PCIe enumeration process. The user application should set the DBI_CS2 bit (bit 5 in the Command Status Register) first, then configure the BAR5 Mask Register and clear the DBI_CS5 bit after configuration. Do not attempt to modify the BAR Mask Registers from the serial link side (from RC or host). For more information, see the *PCIe for KeyStone Devices User's Guide* (SPRUGS6).

Usage Note 16 *Sticky Bits in PCIe MMRs Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: The sticky bits in PCIe memory-mapped registers (MMRs) are those bits that are neither initialized nor modified by a hot reset, as defined in the PCI Express base specification. The values of the sticky bits are unchanged after a hot reset, which is important in error handling to ensure that error-related control and status information is not lost due to a hot reset.

In the C6672 device, the sticky bits in PCIe MMRs are not changed after a soft reset; only the non-sticky bits are initialized. During a hard reset, both sticky and non-sticky bits get reset. For more information, see the PCI Express base specification.

Usage Note 17 *The Clock Input to NETCP Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: The clock input to the Network Coprocessor (NETCP) is programmable. A multiplexer selects between SYSCLK1 or the output of PASS PLL as the input to NETCP. The multiplexer is controlled by bit 13 (zero indexed) in the PASSPLLCTL1 register. The default value of this bit is 0 which selects SYSCLK1 as the input to NETCP; however, this is not the recommended mode of operation.

Workaround: In order to set the PASS PLL output as the input to NETCP, bit 13 should be set to 1. This can be done as part of the PASS PLL initialization sequence. Read-modify-write can be used to make sure other bits in the register are not affected. Note that during Ethernet boot, the Boot ROM code correctly sets this bit to 1. However, in all other boot modes this bit has a default value of 0 and the software must change the value to 1.

Usage Note 18 *Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: During the power-up and power-down cycles of the DSP, it is possible that some current may flow between the VDDR and VDDT rails. The VDDR rail tracks the VDDT rail as VDDT is ramping up to 1.0 V. When VDDR gets approximately to the 400 mV mark, the VDDR rail stops tracking the VDDT rail. Leakage observed here comes from the SerDes module that contains 1 V transistors. It has been verified that this leakage is expected and has no impact on the reliability of the device.

Usage Note 19 *CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: A multi-rank configuration presents certain limitations when using lower speed bins like DDR3-800 that require CWL = 5 to be programmed.

It is recommended that the following guidelines be observed for multi-rank configuration when using CWL=5:

- Set bit 24 (indexed to 0) in DDR3_CONFIG_REG_12 at 0x02620434 to 1. This will constrain the DDR PHY to use only rank0 delays for reads/writes to both ranks. The reset value is 0, which allows each rank to use its own delay.
- Using only rank0 delays means that the fly-by/round-trip delay across ranks must be balanced for a given byte lane.
- Because rank0 delays are used by the PHY for both ranks, only the single rank equations for fly-by and round-trip delays from *DDR3 Design Requirements for KeyStone Devices* (SPRABI1) need to be satisfied. The multi-rank equations will not be valid.

Usage Note 20 *IDMA1 Performance Limitation Usage Note*

Revision(s) Affected: 1.0

Details: In silicon revision 1.0, the IDMA1 transaction performance is lower than the theoretical number due to the insufficient buffering in the EMC. This issue has been fixed in silicon revision 2.0 and the performance meets the theoretical number. The following table shows the realistic IDMA1 transaction performance.

Table 10. IDMA1 Transaction Performance

IDMA1 Transaction	Silicon Revision 1.0 (Bytes/Cycle)	Silicon Revision 2.0 (Bytes/Cycle)	Theoretical Performance (Bytes/Cycle)
L1D to L2	4.54	7.69	8
L2 to L1D	3.41	7.67	8
L2 to L2	3.75	7.76	8
L1D Fill	2.66	7.84	8
L2 Fill	7.49	15.44	16

Usage Note 21 *Revised PLL Programming Sequence Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: It has been observed that on a few devices, the CorePacs lock up after reprogramming of the Core PLL. It has been identified that the incorrect PLL programming sequence was causing the CorePacs to lock up.

Workaround: TI has revised the PLL programming sequence for Main PLL, DDR PLL and PASS PLL to eliminate the possibility of this lock-up issue. The revised sequence enables the by-pass mode in the PLL Controller via a MUX (by clearing PLLEN and PLENSRC bits) when the Main PLL is being re-programmed. Also, the PLLM, PLLD and BWADJ fields are programmed prior to assertion of PLLRST signal for all three PLLs in the revised sequence.

Please use the revised Main PLL, DDR PLL and PASS PLL programming sequence as described in *KeyStone Architecture Phase-Locked Loop (PLL) User Guide* (SPRUGV2).

Usage Note 22 *Core Wake Up on RESET Usage Note*

Revision(s) Affected: 2.0

Details:

Execution may start only on some C66x 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 C66x 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 C66x CorePacs to go into an unknown state and they don't start execution.

Resets using POR and RESETFULL does 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 RESET pin. This behavior can only happen in the lab environment where 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 C66x CorePacs
2. Disconnect all the C66x 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 C66x CorePacs to wake up correctly on reset via RESET. Bypassing either step 1 or 2 will result in C66x CorePacs that do not begin execution after reset.

Usage Note 23 *BSDL Testing Support Usage Note*

Revision(s) Affected: 1.0, 2.0

Details:

It has been observed that IEEE1149.6 testing on the device may not function properly. During AC boundary scan testing, errors (either stuck at or shorts) may be reported by the BSDL test software and BSDL controller.

Proper use of the SerDes AC boundary scan cells requires that the SoC (processor) be powered with clocks supplied in the recommended sequencing as outlined in the data manual for the device in use, and the device is out of reset prior to running the boundary scan tests.

Additionally, it has been identified that the default SerDes RX termination value, “RXTERM”, is set to either 000 or 111 and must be re-programmed to 001 (0.7VDDT (or 0.8VDDT) common point in the memory mapped registers (MMR)) for each SerDes prior to use.

This correction has been found to be valid for all SerDes except PCIe where the register termination control had been hard coded and is not configurable except through bit 5 (pcs_fix_term) of the PCS Configuration 0 Register [PCS_CFG0]. Configuring bit 5 high forces the termination value to be set to common point to vsst; setting this MMR bit to a “0” sets the termination value to common point floating. In either case the boundary scan cell may not function properly.

Workaround:

The only known workaround involves correctly powering and providing clocking for the SoC/DSP (processor) first. Each SerDes RX termination (RXTERM) register will then be required to be re-programmed from its default value of “000” with a value of “001” prior to BSDL testing. Depending on the SerDes peripheral, there may be more than one RXTERM register requiring programming.

There are two types of BSDL tools currently available:

1. Those capable of running a BSDL controller and TI emulation/debugger software over the same hardware platform
2. Those types of tools that are designed to run emulation/debugger software independently from boundary scan software

The following are the known workaround steps for both scenarios described above:

Single hardware/software tool

1. The SoC/DSP must be correctly powered with clocks applied
2. Using the emulation/debugger, enable all the respective SerDes peripherals
3. Using the emulation/debugger, change the RXTERM value from “000” to “001”
4. Disable the emulation/debugger software and connect using your BSDL software
5. Run your respective 1149.6 boundary scan tests

Separate hardware/software tool

1. The SoC/DSP must be correctly powered with clocks applied
2. Using the emulation/debugger, enable all the respective SerDes peripherals
3. Using the emulation/debugger, change the RXTERM value from “000” to “001”
4. Disconnect the emulation/debugger hardware and software
5. Connect the boundary scan hardware and software
6. Run your respective 1149.6 boundary scan tests

See the indicated peripheral user’s guide for details on enabling a given peripheral.

Table 11. Register Information for the Peripheral

Peripheral	Register	Bit	Address
HyperLink (SPRUGW8)	HYPERLINK_SERDES_CFGRX0	9:7	0x026203B8
	HYPERLINK_SERDES_CFGRX1	9:7	0x026203C0
	HYPERLINK_SERDES_CFGRX2	9:7	0x026203C8
	HYPERLINK_SERDES_CFGRX3	9:7	0x026203D0
PCIe (SPRUGS6)	PCS_CFG0 (RXTERM value not configurable to 001)	5	0x21800380
SGMII (SPRUGV9)	SGMII_SERDES_CFGRX0	9:7	0x02620344
	SGMII_SERDES_CFGRX1	9:7	0x0262034C
SRIO (SPRUGW1)	SRIO_SERDES_CFGRX0	9:7	0x02620364
	SRIO_SERDES_CFGRX1	9:7	0x0262036C
	SRIO_SERDES_CFGRX2	9:7	0x02620374
	SRIO_SERDES_CFGRX3	9:7	0x0262037C

Usage Note 24 *Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: Users are required to program their board CVDD supply initial value to 1.1V on the device. The initial CVDD voltage at power-on will be 1.1V 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 I Devices* (SPRABI2) is absolutely required.

Usage Note 25 *Performance Degradation for Asynchronous Access Caused by An Unused Feature Enabled in EMIF16 Usage Note*

Revision(s) Affected: 1.0, 2.0

Details: Although it supports only asynchronous mode operation on the device, the EMIF16 module has a legacy 'synchronous mode' feature that is enabled by default. While this synchronous mode is enabled, EMIF16 issues periodic refresh commands that take precedence over asynchronous accesses commands and stall the execution of the latter until the refresh command is executed. This stall results in reduced throughput of asynchronous accesses when EMIF16 tries to read or write to the asynchronous memory. The stall will manifest itself as a long delay between asynchronous accesses.

Workaround: Programming bit 31 at the 32-bit address 0x20C00008 to 1 will disable the synchronous mode feature.

```
*(Uint32*) 0x20C00008 |= 0x80000000; //Disable synchronous mode feature
```

When the synchronous mode is disabled, EMIF16 will not issue any refresh commands. This will no longer result in stall cycles between asynchronous accesses and thus the performance will be improved.

This bit affects only the refreshes issued by EMIF16. It does not affect the rest of the device.

Usage Note 26 *DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note*

Revision(s) Affected: 1.0, 2.0

Summary: For commands with a higher class of service, larger than expected latency may be observed due to the DDR3 memory controller failing to properly elevate the priority of execution of the command.

Details: The DDR3 memory controller's 'Class of service' (COS) feature allows the user to prioritize commands that are scheduled inside the controller's command FIFO based on bus priority of a master or its master ID. A latency counter is programmed for a specific class of service. The counter value decides how long the command may wait inside the FIFO before it is moved to head of the FIFO. The commands assigned to a class of service with a lower counter value are considered to have a higher class of service. Please see the *KeyStone Architecture DDR3 Memory Controller User Guide* (SPRUGV8) for details.

The following are examples of how read and write transactions behave when the class of service feature is enabled. The class of service for reads and writes are handled independently, i.e. if the counter for a read with a higher class of service expires, all writes irrespective of their class of service will not be blocked. All other reads with a lower class of service, however, will be blocked. Similarly, if the counter for a write with a higher class of service expires, all reads irrespective of their class of service will not be blocked. All other writes with a lower class of service, however, will be blocked. This can lead to a situation where the controller fails to elevate the priority of execution of a command that has been assigned to a higher class of service.

An example of such a situation is when the COS counter for a read command with a higher class of service expires and there is a continuous stream of write traffic to an open bank in the memory with lower class of service than the read command.

NOTE: The class of service counter or class of service latency tracks how long a particular command that is mapped to the specific COS (in this example, the read command) should wait in the FIFO before it is prioritized for execution.

NOTE: The PR_OLD_COUNT tracks how long the oldest command (regardless of the assigned COS) should stay in the command FIFO before it is prioritized for execution.

In the above example, the COS counter associated with that read command expires. However, since the controller will always prioritize writes to an open bank, the read command will become the oldest in the FIFO. From this point, the PR_OLD_COUNT starts tracking how long it has been in the FIFO. It is only when PR_OLD_COUNT expires that the read command will be moved to the head of the FIFO and executed. Thus, commands with a higher class of service may see a higher than expected latency of execution which can be a problem if a master is latency sensitive.

NOTE: The situation that exposes this issue (like the one mentioned above) is expected to be a corner case. In a real world system that has random traffic generated by multiple masters, the probability of this issue is very slim.

NOTE: This does not affect systems where the class of service feature is not used.

Workaround:

The PR_OLD_COUNT field in the VBUSM configuration register decides the number of DDR3 clock cycles after which the controller raises the priority of the oldest command in the FIFO. By default, this is 0xFF which translates to 0xFF x 16 x 2 DDR3 clock cycles (refer to the *KeyStone Architecture DDR3 Memory Controller User Guide* (SPRUGV8)). This field should be reduced until the higher than expected latency for the latency sensitive master is reduced to an acceptable level. The time spent by the oldest command in the FIFO depends on the traffic pattern and the aggregate traffic hitting the DDR interface. Thus, there is no optimal value that can be recommended for all systems. The user is expected to determine the optimal PR_OLD_COUNT.

Potential effect of reducing PR_OLD_COUNT:

The controller attempts to manage the traffic to/from DDR as efficiently as possible by rescheduling commands in the FIFO as per its arbitration logic. The user should note that while reducing PR_OLD_COUNT too low (0x0 to 0x20) will reduce the time spent by a command inside the FIFO, this may impact the scheduling and negatively affect throughput.

Revision History

Changes from May 15, 2014 to April 30, 2015 (from G Revision (May 2014) to H Revision) **Page**

- Changed from: The lower 8 bits (bits 7:0) may be written with a read data eye sample value or left at their default value of 0x34. to: The lower 8 bits (bits 7:0) must not be overwritten and will contain their default value of 0x34. 19
- Added Limited support is available for this solution. The customer will need to assume responsibility for validating all required timing margins are met. 19
- Added This solution is functional on standard DDR3 fly-by layouts and is referred to as Full Automatic Leveling to workaround 3. 20
- Added Advisory 29, ***False DDR3 Write ECC Error Reported Under Certain Conditions*** 47
- Updated NOTE 2 Advisory 29, ***False DDR3 Write ECC Error Reported Under Certain Conditions*** 47
- Updated Workaround 2 QA Advisory 29, ***False DDR3 Write ECC Error Reported Under Certain Conditions*** 48
- Added Advisory 30, ***Descriptors Placed in PCIe Memory Space can Cause Problems*** 49

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