

TMS320C6452 Digital Media Processor

Silicon Errata



Literature Number: SPRZ279B
June 2008–Revised June 2009

1	Introduction	5
1.1	Device and Development Support Tool Nomenclature	5
1.2	Revision Identification	6
2	Silicon Revision 1.1 Usage Notes and Known Design Exceptions to Functional Specifications	7
2.1	Usage Notes for Silicon Revision 1.1	7
2.1.1	PCI Cannot Burst More Than 64 bytes When Used in Master Mode (Silicon Rev. 1.0 and 1.1) ..	7
2.1.2	DDR2 Addressing Range Difference (Silicon Rev. 1.0 and 1.1)	7
2.1.3	The 3-Port Ethernet Switch Subsystem (3PSW) Host DMA Controller Data Traffic Can Cause Deadlock Situation (Silicon Rev. 1.0 and 1.1)	7
2.2	Silicon Revision 1.1 Known Design Exceptions to Functional Specifications	8
3	Silicon Revision 1.0 Usage Notes and Known Design Exceptions to Functional Specifications	21
3.1	Usage Notes for Silicon Revision 1.0	21
3.2	Silicon Revision 1.0 Known Design Exceptions to Functional Specifications	21
	Appendix A Revision History	23

List of Figures

1	Sample Revision Diagram	6
2	IDMA, SDMA, MDMA Paths	11

List of Tables

1	Silicon Revisions	6
2	Addressing Range Differences.....	7
3	Silicon Revision 1.1 Advisory List	8
4	Silicon Revision 1.0 Advisory List.....	21
A-1	Changes Made in Revision B.....	23

TMS320C6452 DSP

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320C6452. The updates are applicable to the ZUT package.

For additional information, see the *TMS320C6452 Digital Signal Processors Data Manual* ([SPRS371](#)).

1.1 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all DSP devices and support tools. Each DSP commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g., TMS320C6452). Texas Instruments recommends two of three 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.

1.2 Revision Identification

Figure 1 provides an example of the TMS320CC6452 and device markings. The device revision can be determined by the symbols marked on the top of the package. Some prototype devices may have markings different from those illustrated.

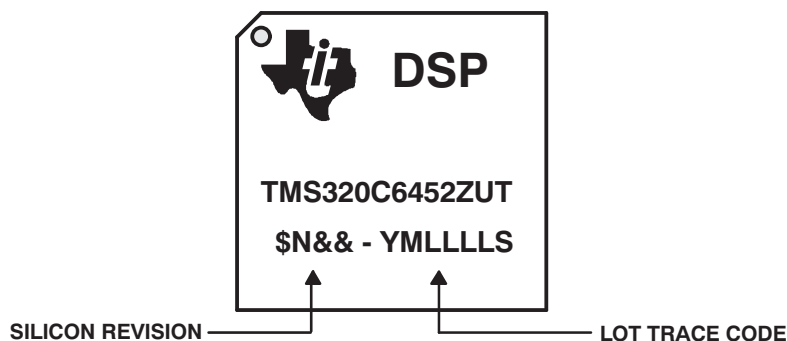


Figure 1. Sample Revision Diagram

Table 1. Silicon Revisions

SILICON REVISION	SILICON REVISION MARKING (Value of &&)	CPU REVISION	C64X+ MEGAMODULE REVISION	JTAG ID REGISTER VALUE	ORDERABLE PART NUMBER
1.0	10	1.0 (CPU ID = 10h REVISION_ID = 00h)	Rev. 3 (MM_REVID[REVISION] = 2h)	0x0B77 A02F	TMS320C6452ZUT9/7
1.1	11	1.0 (CPU ID = 10h REVISION_ID = 00h)	Rev. 3 (MM_REVID[REVISION] = 2h)	0x1B77 A02F	TMS320C6452ZUT9/7

2 Silicon Revision 1.1 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 1.1 of the C6452 devices.

2.1 Usage Notes for Silicon Revision 1.1

Usage notes highlight and describe particular situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness. These usage notes will be incorporated into future documentation updates for the device (such as the device-specific data sheet), and the behaviors they describe will not be altered in future silicon revisions.

2.1.1 PCI Cannot Burst More Than 64 bytes When Used in Master Mode (Silicon Rev. 1.0 and 1.1)

On all C6452 silicon revisions, the PCI on the device can operate as a PCI master and slave. As a slave, the device PCI responds to accesses initiated by an off-chip PCI master. As a master, the C6452 PCI initiates transfers on the PCI bus. Usually, for memory read and write transfers, another C6452 master such as the EDMA is configured to move data to/from the PCI.

As a PCI master, the C6452 PCI is capable of bursting only a maximum of up to 64 bytes. In other words, for memory transfers larger than 64 bytes, the PCI initiates a transfer, transfers 64 bytes, stops the transfer, and then repeats. As a PCI slave, external PCI masters can burst an infinite amount of data to the PCI. Note that the PCI may insert wait states or generate a target retry if it cannot meet the latency requirement set forth by the PCI system. For example, a PCI access to DDR2 memory may stall due to other master accesses or because of a scheduled DDR2 memory refresh. In this case, the PCI generates a target retry until the DDR2 memory controller is ready to service the PCI request.

Because of this limitation, the PCI throughput will be lower in master mode than in slave mode. To avoid low throughput performance, external PCI masters should be used to move data to/from the PCI whenever possible.

2.1.2 DDR2 Addressing Range Difference (Silicon Rev. 1.0 and 1.1)

On silicon revision 1.1, the DDR2 addressing range has been increased from 256M Bytes to 512M Bytes. [Table 2](#) shows the differences between the two revisions.

Table 2. Addressing Range Differences

SILICON REVISION	START ADDRESS	END ADDRESS	SIZE (Bytes)	C64x+ MEMORY MAP
1.1	0xE000 0000	0xFFFF FFFF	512M	DDR2 SDRAM
1.0	0xE000 0000	0xEFFF FFFF	256M	DDR2 SDRAM
				(address range 0xF000 0000 - 0xFFFF FFFF is reserved in silicon revision 1.0)

2.1.3 The 3-Port Ethernet Switch Subsystem (3PSW) Host DMA Controller Data Traffic Can Cause Deadlock Situation (Silicon Rev. 1.0 and 1.1)

On silicon revision 1.0/1.1, the user must be sure that there is no ongoing traffic from the 3-Port Ethernet Switch Subsystem (3PSW) Host DMA Controller to the DSP internal memory or DDR2 when the RESET is asserted. Otherwise, the chip may end up in a deadlock situation where a POR has to be asserted.

2.2 Silicon Revision 1.1 Known Design Exceptions to Functional Specifications

This section contains the most recent advisories, including those that have been fixed in the Silicon Rev. 1.1.

Table 3. Silicon Revision 1.1 Advisory List

Advisory — Description	Page
Advisory 1.1.1 PCI Device ID has been corrected	9
Advisory 1.1.2 The 3-Port Ethernet Switch Subsystem (3PSW) clocking problem in normal functional operation	9
Advisory 1.1.3 SDMA/IDMA: When DSP Level 2 memory is configured as non-cache (RAM), unexpected blocking and potential deadlock condition may occur	10
Advisory 1.1.4 SerDes clocking problem in normal functional operation	15
Advisory 1.1.5 Processor hangs, if a $\overline{\text{POR}}$ is applied	16
Advisory 1.1.6 L19 pin always shows a voltage of 0 V	17
Advisory 1.1.7 Expiration of watchdog timer (WDT) does not reset the device	18
Advisory 1.1.8 DMA Access to L2 SRAM May Stall When the DMA Has Lower Priority Than the CPU	19
Advisory 1.1.9 DMA Access to L2 SRAM May Stall When the DMA and the CPU Command Priority is Equal	20

Advisory 1.1.1***PCI Device ID has been corrected***

Revision(s) Affected

1.1

Details

Advisory 1.0.1 of Silicon Revision 1.0 has been fixed in Silicon Revision 1.1. The DEV_ID field of PCIVENDEV register (offset 0x000) has been changed to 0xB003. See [Section 3.2](#) for details applicable to Silicon Rev. 1.0.

Advisory 1.1.2***The 3-Port Ethernet Switch Subsystem (3PSW) clocking problem in normal functional operation***

Revision(s) Affected

1.1 and earlier

Details

TCK (JTAG controller clock) is internally shared by the 3-Port Ethernet Switch Subsystem's (3PSW) STCICLK for test and debug mode. In order for the 3-Port Ethernet Switch Subsystem (3PSW) to get proper clocking in the normal functional operation, STCICLK needs to be held low. But since there is an internal pullup on TCK, it keeps the 3-Port Ethernet Switch Subsystem from locking to external REFCLKP/N for proper operation.

Workaround(s)

The TCK pin should be externally pulled down with a 1-k Ω resistor.

Advisory 1.1.3
SDMA/IDMA: When DSP Level 2 memory is configured as non-cache (RAM), unexpected blocking and potential deadlock condition may occur
Revision(s) Affected

1.1 and earlier

Details

Note: This advisory is not applicable if DSP L2 memory is configured as 100% cache or L2 RAM is not accessed by IDMA or SDMA during run-time.

The C64x+ megamodule has a master direct memory access (MDMA) bus interface and a slave direct memory access (SDMA) bus interface. The MDMA interface provides DSP access to resources outside the C64x+ megamodule (i.e., DDR2, EMIFA, VLYNQ remote memory).

The MDMA interface is typically used for CPU/cache accesses to memory beyond the level-2 (L2) memory level. These accesses include cache line allocates, write-backs, and non-cacheable loads and stores to/from system memories.

The SDMA interface allows other master peripherals, including EDMA transfer controllers, 3PSW, HPI, PCI, and VLYNQ, to access level-1 data (L1D), level-1 program (L1P), and L2 RAM DSP memories.

The DSP internal DMA (IDMA) is a C64x+ megamodule DMA engine used to move data between internal DSP memories (L1, L2) and/or the DSP peripheral configuration bus. The IDMA engine shares resources with the SDMA interface.

The C64x+ megamodule has an L1D cache and L2 cache both implementing write-back data caches. It holds updated values for external memory as long as possible. It writes these updated values, called "victims," to external memory when it needs to make room for new data or when requested to do so by the application. The L1D sends its victims to L2. The caching architecture has pipelining, meaning multiple requests could be pending between L1, L2, and MDMA. For more details on the C64x+ megamodule and its MDMA and SDMA ports, see the *TMS320C64x+ Megamodule Reference Guide* (literature number [SPRU871](#)).

Ideally, the MDMA (dotted line in [Figure 2](#)) and SDMA/IDMA paths (dashed lines in [Figure 2](#)) operate independently with minimal interference. Normally, MDMA accesses may stall for extended periods of time due to expected system level delays (e.g., bandwidth limitations, DDR2 memory refreshes). However, when using L2 as RAM, SDMA and IDMA accesses to L2/L1 may experience unexpected stalling in addition to the normal stalls seen by the MDMA interface. For latency-sensitive traffic, the SDMA stall can result in missing real-time deadlines. In a more severe case, the SDMA stall can produce a deadlock condition in the device. An IDMA stall cannot produce a deadlock condition.

Note: SDMA/IDMA accesses to L1P/D will not experience an unexpected stall if there are no SDMA/IDMA accesses to L2. Unexpected SDMA/IDMA stalls to L1 happen only when they are pipelined behind L2 accesses. Additionally, the deadlock scenario will be avoided if there are no SDMA accesses to L2.

[Figure 2](#) is provided for illustrative purposes and is incomplete for simplification. The SDMA/IDMA (dashed lines) path could also go to L1D/L1P memories and IDMA can go to DSP CFG peripherals. MDMA transactions can also originate from L1P or L1D through the L2 controller or directly from DSP.

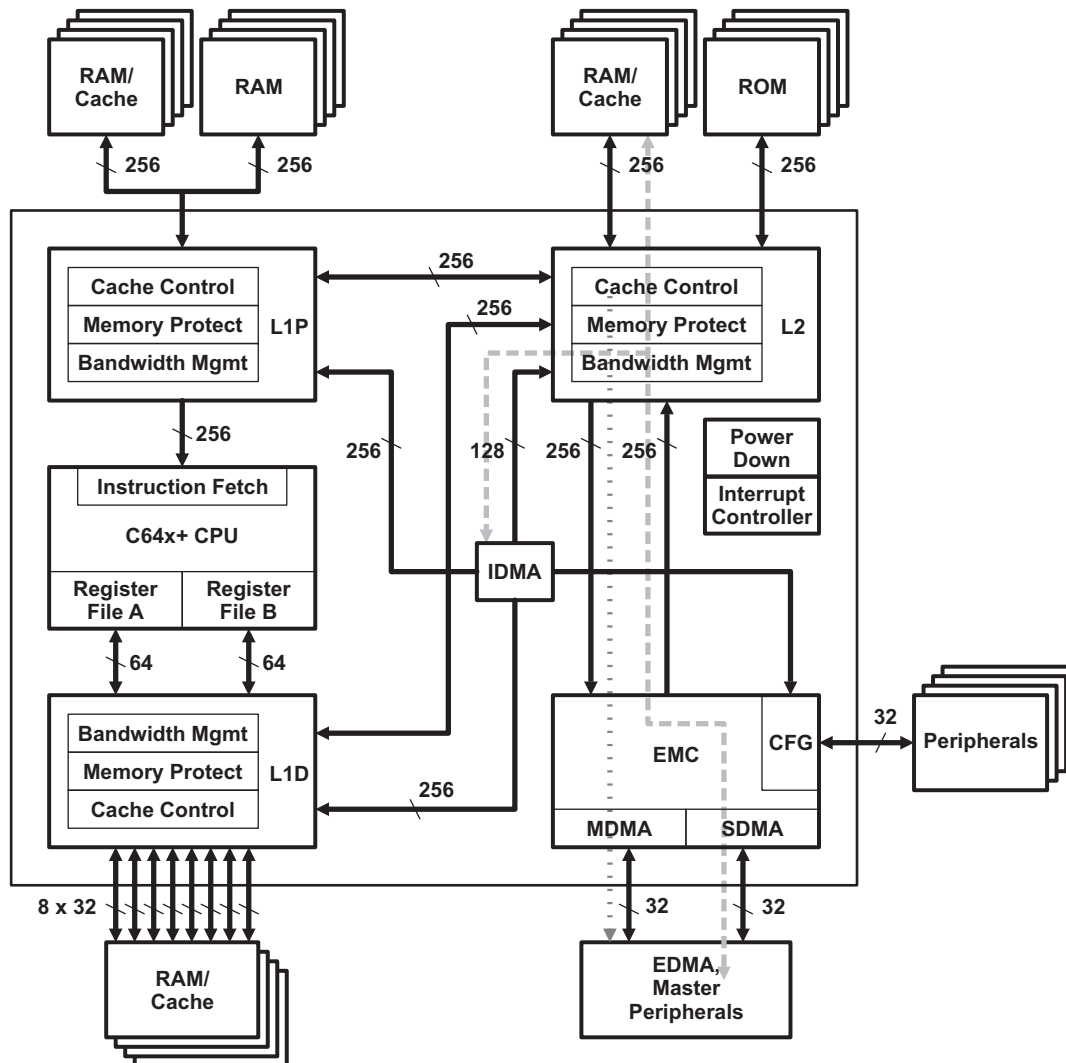


Figure 2. IDMA, SDMA, MDMA Paths

SDMA/IDMA stalls may occur during the following scenarios. Each of these scenarios describes expected normal DSP functionality, but the SDMA/IDMA access potentially exhibits additional unexpected stalling.

1. Bursts of writes to non-cacheable MDMA space (i.e., DDR2, EMIFA, VLYNQ remote). The DSP buffers up to four non-cacheable writes. When this buffer fills, SDMA/IDMA is blocked until the buffer is no longer full. Therefore, bursts of non-cacheable writes longer than three writes can stall SDMA/IDMA traffic.
2. Various combinations of L1 and L2 cache activity
 - a. L1D read miss generating victim traffic to L2 (cache or SRAM) or external memory. The SDMA/IDMA may be stalled while servicing the read miss and the victim. If the read miss also misses L2 cache, the SDMA/IDMA may be stalled until data is fetched from external memory to service the read miss.
 - b. L1D read request missing L2 (going external) while another L1D request is pending. The SDMA/IDMA may be stalled until the external memory access is complete.
 - c. L2 victim traffic to external memory during any pending L1D request. SDMA/IDMA may be stalled until external memory access and the pending L1D request is complete.

The duration of the SDMA/IDMA stalls depends on the quantity/characteristics of the L1/L2 cache and MDMA traffic in the system. In cases 2a, 2b, and 2c above, stalling may or may not occur depending on the state of the cache request pipelines and the traffic target locations. These stalling mechanisms may also interact in various ways, causing longer stalls. Therefore, it is difficult to predict if, and how long, stalling will occur.

SDMA/IDMA stalling and any system impact is most likely in systems with excessive context switching, L1/L2 cache miss/victim traffic, and heavily loaded DDR2 interface.

Use the following procedure to determine if SDMA/IDMA stalling is the cause of real-time deadline misses for existing applications. Situations where real-time deadlines may be missed include loss of McASP samples and low peripheral throughput.

1. Determine if the transfer missing the real-time deadline is accessing L2 or L1D memory. If not, then SDMA/IDMA stalling is not the source of the real-time deadline miss.
2. Identify all SDMA transfers to/from L2 memory (e.g., EDMA transfer to/from L2 from/to a McASP, HPI block transfer to/from L2). If there are no SDMA transfers going into L2, then SDMA/IDMA stalling is not the source of the problem.
3. Redirect all SDMA transfers to L2 memory to other memories using one of the following methods:
 - a. Temporarily transfer all the L2 SDMA transfers to L1D SRAM.
 - b. If not all L2 SDMA transfers can be moved to L1D memory, temporarily direct some of the transfers to DDR memory and keep the rest in L1D memory. Still, there should be no L2 SDMA transfers.
 - c. If Steps a and b are not possible, move the transfer with the real-time deadline to 3PSW CPPI RAM. If the EMAC CPPI RAM is not big enough, a two-step mechanism can be used to page a small working buffer defined in 3PSW CPPI RAM into a bigger buffer in L2 SRAM. The EDMA module can be setup to automate this double-buffering scheme without CPU intervention for moving data from 3PSW CPPI RAM. Some throughput degradation is expected when the buffers are moved to 3PSW CPPI RAM.

Note: 3PSW CPPI RAM memory is only word-addressable. Therefore, EDMA transfers to/from 3PSW CPPI RAM must have SRCBIDX/DSTBIDX = 4. If real-time deadlines are still missed after implementing any of the options in Step 3 above, then SDMA/IDMA stalling is likely not the cause of the problem. If real-time deadline misses are solved using any of the options in Step 3, then SDMA/IDMA stalling is likely the source of the problem.

As previously mentioned, a possible deadlock scenario is introduced in the presence of the SDMA stalls just described. This scenario occurs for certain masters connected to the data SCR, either directly or indirectly, through a bridge. For the C6452 device, the masters that fall into this category are the EDMA transfer controllers (EDMA TCs), PCI, UHPI, VLYNQ, and 3PSW. If the following sequence of events occurs, then a deadlock situation might arise:

1. One of the following three accesses occur:
 - a. An EDMA TC issues a write command to the DSP's SDMA followed by a subsequent write command to DDR2 (or EMIFA).
 - b. HPI, PCI, or VLYNQ issues a write command to the DSP's SDMA and HPI, PCI, or VLYNQ issues a subsequent write command to DDR2 (or EMIFA).
 - c. The 3PSW issues a write command to the DSP's SDMA followed by a subsequent write command to DDR2 (or EMIFA).
2. The DSP's SDMA asserts itself not ready and is unable to accept more write data and a cache line writeback is initiated from DSP memory to DDR2 memory (or another slave memory, e.g., EMIFA).

In the above scenario, it is possible for data phases from the write command issued to

DDR2 (or EMIFA) to be stuck behind the data phases for the write to the DSP's SDMA in the SCR.

Therefore, if the DSP issues victim traffic to the same slave (DDR2 or EMIFA), then data associated with the victim traffic (#2) intended for DDR2 (or EMIFA) will be stuck behind write commands issued for #1. However, due to the MDMA/SDMA blocking issue, the SDMA traffic for #1 will be waiting for the MDMA traffic for #2 to finish, manifesting itself into a deadlock situation.

Workarounds

Method 1

Entirely eliminate SDMA/IDMA stalling and potential for a deadlock condition using one, or both, of the following methods:

1. Configure entire L2 RAM as 100% cache (e.g., move all data buffers to L1D/P, 3PSW CPPI memory, or external memory).

Note: Some throughput degradation is expected when the buffers are moved to EMAC CPPI RAM. CPPI memory is only word-addressable. Therefore, EDMA transfers to/from EMAC CPPI RAM must have SRCBIDX/DSTBIDX = 4.

2. Eliminate all SDMA/IDMA access to L2 RAM during any time SDMA/IDMA stalling would have an impact (e.g., could preload data/code through SDMA/IDMA during system initialization/re-configuration).

Method 2

To reduce the SDMA/IDMA stalling system impact, perform any of the following:

1. Improve system tolerance on the DMA side (SDMA/IDMA/MDMA):
 - a. Understand and minimize latency-critical SDMA/IDMA accesses to L2 or L1P/D.
 - b. Directly reduce critical real-time deadlines, if possible, at peripheral/IO level (e.g., increase word size and/or reduce bit rates on serial ports).
 - c. To reduce DSP MDMA latency:
 - i. Increase the priority of DSP access to DDR2/EMIFA such that MDMA latency of MDMA accesses causing stalls is minimized.

Note: Other masters such as HPI may have real-time deadlines which dictate higher priority than DSP.

- ii. Lower the PR_OLD_COUNT field setting in the DDR2 memory controller's burst priority register. Values ranging between 0x10 and 0x20 should give decent performance and minimize latency; lower values may cause excessive SDRAM row thrashing.
- iii. Do not perform EMIFA access using EMIFA WAIT handshaking during DSP run-time. Devices using WAIT potentially insert excessive latency to external memory accesses.
2. Minimize offending scenarios on DSP/Caching side:
 - a. If DSP performing non-cacheable writes is causing the issue, insert non-cacheable reads every few writes to allow write buffer to drain.
 - b. Avoid caching from slow memories, such as asynchronous memory. Instead, page the data via the EDMA from the off-chip async memory to L2 SRAM or SDRAM space before accessing the data from the DSP.

Note: Paging cannot occur while real-time deadlines must be met.

- c. Use explicit cache commands to trigger cache write-backs during appropriate times (L1D writeback all, L2 writeback all). Do not use these commands when real-time deadlines must be met.

- d. Restructure program data and data flow to minimize the offending cache activity.
 - i. Define the read-only data as "const." The const C keyword tells the compiler that the array will not be written to. By default, such arrays are allocated to the ".const" section as opposed to BSS. With a suitable linker command file, the developer can link the .const section off-chip, while linking .bss on-chip. Because programs initialize .bss at run time, this reduces the program's initialization time and total memory image.
 - ii. Explicitly allocate look-up tables and writeable buffers to their own sections. The #pragma DATA_SECTION(label, "section") directive tells the compiler to place a particular variable in the specified COFF section. The developer can explicitly control the layout of the program with this directive and an appropriate linker command file.
 - iii. Avoid directly accessing data in slow memories (e.g., flash); copy at initialization time to faster memories.
- e. Modify troublesome code.
 - i. Rewrite using DMAs to minimize data cache writebacks. If the code accesses a large quantity of data externally, consider using DMAs to bring in the data, using double buffering and related techniques. This will minimize cache write-back traffic and the likelihood of SDMA/IDMA stalling.
 - ii. Re-block the loops. In some cases, restructuring loops can increase reuse in the cache and reduce the total traffic to external memory.
 - iii. Throttle the loops. If restructuring the code is impractical, then it is reasonable to slow it down. This reduces the likelihood that consecutive SDMA/IDMA blocks "stack up" in the cache request pipelines resulting in a long stall.

To eliminate the potential for a deadlock condition, perform the following:

1. Force each EDMA TC to perform writes to either DSP memory space or DDR2 (or EMIFA) memory space, but not to both.
2. For 3PSW, VLYNQ, HPI, and PCI, do one of the following:
 - a. Force the completion of pending write commands to either DSP memory space or DDR2/EMIFA memory space before initiating writes to a different destination. Pending write commands from a particular master are forced to complete when the same master initiates a read from the same destination memory. Note that in the case of DDR2 and EMIFA, a read command only forces the completion of write commands within a 2KB-aligned window.
 - b. Force each master to perform writes to either DSP memory space or DDR2 (or EMIFA) memory space, but not to both.

Note: In the case of VLYNQ, PCI, and HPI, as a group, these masters must only perform writes to either DSP memory or DDR2/EMIFA memory, but not to both. For example, if VLYNQ writes to DSP memory and PCI writes to DDR2 memory, the potential for the deadlock condition is still present.

Advisory 1.1.4***SerDes clocking problem in normal functional operation***

Revisions Affected

1.0, 1.1

Details

A bug has been found in the SerDes interfaces which cause SerDes clocking problem in normal functional operation. This problem will not occur when external pulldown is applied on the TCK (JTAG controller clock) pin. SERDES are used in the 3port Ethernet Switch Subsystem (3PSW).

TCK (JTAG controller clock) is internally assigned to the Serial Test and Configuration Interface Clock (STCICLK) that is used by the SerDes macro. In order for the SerDes macro to get proper clocking in the normal functional operation, it needs STCICLK to be held low. But since there is an internal pullup on TCK, it creates problems for SerDes operation.

Workaround

TCK pin should be externally pulled down with a 1-k Ω resistor.

Advisory 1.1.5 ***Processor hangs, if a $\overline{\text{POR}}$ is applied***

Revisions Affected 1.0, 1.1

Details On very rare occasions the C6452 processor hangs, if a $\overline{\text{POR}}$ is applied during CPU processing of a pipelined instruction. This is because the boot loader code contains a read reference from an uninitialized register, B3 and a write to memory reference calculated using an uninitialized register B15. The first time the device boots or on $\overline{\text{POR}}$, B15 value is mostly 0x00000000 or 0xFFFFFFFF. This will not cause hang condition, as 0x00000000 is reserved and 0xFFFFFFFF is located in DDR2 with PLL enabled.

The hang will only happen if the following two conditions are met:

1. CPU is in pipeline such that upon $\overline{\text{POR}}$ B15 is neither 0x0000 0000 nor 0xFFFF FFFF.
2. This address falls in an IP address area which does not have its clock enabled.

Once the above two conditions are met, the first line of code in Boot ROM tries to access B15 and waits indefinitely, which causes a hang condition. A second $\overline{\text{POR}}$ always clears the B15 to 0x0000 0000 or 0xFFFF FFFF and the CPU is out of hang condition. This condition, which is a very rare event, occurs in both versions of silicon.

Workaround A second $\overline{\text{POR}}$ will resolve the issue and bring the device out of reset properly.

Advisory 1.1.6 ***L19 pin always shows a voltage of 0 V***

Revisions Affected 1.0, 1.1**Details** The pin L19's intended purpose is to act as a V_{CCMON} : The monitor pin indicates the voltage on the die, and, therefore, provides the best probe point for voltage monitoring purposes. But in the current version of silicon this pin always shows a voltage of 0 V.**Workarounds** There is no workaround for this problem in the current version of the silicon.

Advisory 1.1.7 ***Expiration of watchdog timer (WDT) does not reset the device***

Revisions Affected 1.0, 1.1**Details** The watchdog timer's intended purpose is to reset the device upon expiration of the WDT counter. However, in the current versions of silicon, the WDT does not reset the device upon WDT counter expiration.**Workarounds** There is no workaround for this problem in the current version of the silicon.

Advisory 1.1.8 ***DMA Access to L2 SRAM May Stall When the DMA Has Lower Priority Than the CPU***

Revision(s) Affected: 1.1, 1.0

Details: The L2 memory controller in the GEM has programmable bandwidth management features that are used to control bandwidth allocation for all requestors. There are two parameters to control this, command priority and arbitration counter MAXWAIT values. Each requestor has a command priority and the requestor with the higher priority wins. However, there are also counters associated with each requestor that track the number of cycles each requestor loses arbitration. When this counter reaches a threshold (MAXWAIT), which is programmed by the user (or default value), the losing requestor gets an arbitration slot and wins for that cycle. There are four such requestors: CPU, DMA (SDMA and IDMA), user cache coherency operation, and global cache coherency. Global-coherency operations are highest priority, while user-coherency operations are lowest priority. However, there is active arbitration done for the CPU and the DMA (SDMA/IDMA) commands. The priority for DMA commands comes from an external master as part of the SDMA command or a programmable register, IDMA1_COUNT, in the GEM for IDMA commands. The priority for CPU accesses to L2 is in a programmable register, CPUARBU, in the GEM.

More details on the bandwidth management feature can be found in the *C64x+ Megamodule Reference Guide* (literature number [SPRU871](#)).

To enable bandwidth management, the L2 memory controller has an internal (non-user visible) counter that counts MAXWAIT every cycle that a DMA command is blocked because of a CPU access. When the internal counter reaches the MAXWAIT threshold, it is supposed to stay saturated at that value and force the DMA access to win arbitration over the CPU. In the case where DMA priority is less than CPU priority, the internal counter does not saturate at the MAXWAIT threshold value. Instead, it wraps around and keeps counting, thereby, giving more bandwidth to the CPU than intended by the MAXWAIT threshold value. The result is that the DMA may lose to the CPU over multiple arbitration cycles. This typically happens when CPU accesses keep the L2 memory controller busy every cycle; for example, a continuous stream of L1D write misses to L2.

Workaround: Set the CPU at a lower priority than the DMA commands to L2. The priority for CPU accesses to L2 is in a programmable register, CPUARBU, in the GEM. However, lowering the CPU priority may impact the performance since CPU accesses to L2 may stall due to DMA accesses, in case of contention.

Advisory 1.1.9 ***DMA Access to L2 SRAM May Stall When the DMA and the CPU Command Priority is Equal***

Revision(s) Affected: 1.1, 1.0

Details: The L2 memory controller in the GEM has programmable bandwidth management features that are used to control bandwidth allocation for all requestors. There are two parameters to control this, command priority and arbitration counter MAXWAIT values. Each requestor has a command priority and the requestor with the higher priority wins. However, there are also counters associated with each requestor that track the number of cycles each requestor loses arbitration. When this counter reaches a threshold (MAXWAIT), which is programmed by the user (or default value), the losing requestor gets an arbitration slot and wins for that cycle. There are four such requestors: CPU, DMA (SDMA and IDMA), user cache coherency operation, and global cache coherence. Global-coherence operations are highest priority, while user-coherence operations are lowest priority. However, there is active arbitration done for the CPU and the DMA (SDMA/IDMA) commands. The priority for DMA commands comes from an external master as part of the SDMA command or a programmable register, IDMA1_COUNT, in the GEM for IDMA commands. The priority for CPU accesses to L2 is in a programmable register, CPUARBU, in the GEM.

More details on the bandwidth management feature can be found in the *C64x+ DSP Megamodule Reference Guide* (literature number [SPRU871](#)).

The L2 memory controller is supposed to give equal bandwidth to the DMA and the CPU, by alternating between the two for arbitration. Instead, the L2 memory controller gives larger bandwidth allocation to the CPU accesses when the DMA and the CPU priorities are same. The CPU commands keep winning arbitration over the DMA as long as there are no other internal conditions (stalls, etc.) that force the DMA to win arbitration. This typically happens when CPU accesses keep the L2 memory controller busy every cycle, hence, the DMAs stall until the stream of CPU accesses completes. For example, if a continuous stream of L1D write misses to L2 keep the L2 memory controller busy every cycle, the DMAs stall for the entire duration of the write miss stream.

Workaround: Set the CPU and the DMA commands to L2 on different priorities.

3 Silicon Revision 1.0 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 1.0 of the C6452 devices.

3.1 Usage Notes for Silicon Revision 1.0

All usage notes for silicon revision 1.0 also apply to silicon revision 1.1. For details, see [Section 2.1, Usage Notes for Silicon Revision 1.1](#)

3.2 Silicon Revision 1.0 Known Design Exceptions to Functional Specifications

Table 4. Silicon Revision 1.0 Advisory List

Title	Page
Advisory 1.0.1 PCI Device ID of C6452 is same as PCI Device ID of DM643x series of DSP	22

Advisory 1.0.1	<i>PCI Device ID of C6452 is same as PCI Device ID of DM643x series of DSP</i>
Revision(s) Affected	1.0
Details	The DEV_ID field of PCIVENDEV register (offset 0x000) is 0xB001 which is same as the DEV_ID of TMS320DM643x series of DSP.
Workaround(s)	If TMS320DM643x and TMS320C6452 DSPs are used in the same system, the DEV_ID field of the PCIVENDEV register is not sufficient to identify the device. In this case, if PCI boot with auto-initialization is not being used, the JTAGID register (address 0x0204 9018) must be used to identify the device.

Appendix A Revision History

This silicon errata revision history highlights the technical changes made in this revision.

Table A-1. Changes Made in Revision B

LOCATION	ADDITIONS, MODIFICATIONS, AND DELETIONS
Section 2.2	Silicon Revision 1.1 Known Design Exceptions to Functional Specifications: Updated Advisory 1.1.3 , SDMA/IDMA: When DSP Level 2 memory is configured as non-cache (RAM), unexpected blocking and potential deadlock condition may occur

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products

Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
RF/IF and ZigBee® Solutions	www.ti.com/lprf

Applications

Audio	www.ti.com/audio
Automotive	www.ti.com/automotive
Broadband	www.ti.com/broadband
Digital Control	www.ti.com/digitalcontrol
Medical	www.ti.com/medical
Military	www.ti.com/military
Optical Networking	www.ti.com/opticalnetwork
Security	www.ti.com/security
Telephony	www.ti.com/telephony
Video & Imaging	www.ti.com/video
Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2009, Texas Instruments Incorporated