TECHNICAL BULLETIN

TB-89

Differences Between the GT-6426xA and the GT-6426xB

Introduction

This technical bulletin details the feature additions and changes from the GT-6426xA to the new GT-64260B part.

The features primarily addressed are:

- Enhanced CPU Synchronization barrier implementation.
- CPU to PCI ordering.
- Preventing PCI deadlock mechanism.
- Altered Transaction flow control and ARTRY signal assertion.
- Support for Multi-CPU and Symmetric Multi Processing (SMP).

Use this document in coordination with the GT-64260A datasheet available from the Marvell secure Web site.

1. Enhanced CPU Synchronization Barriers

Note

Use this new, enhanced CPU sync barrier feature when cache coherency is enabled. With the synchronization mechanism used in the GT-6426xA (see the CPU Synchronization Barrier section in the GT-64260A datasheet), synchronization is not guaranteed and the result may be a deadlock condition.

In contrast to the GT-6426xA, the GT-64260B implements two sets of sync barrier register. Each set contains a CPU Sync Barrier Trigger register and a CPU Sync Barrier Virtual register (See Section 8.1 "CPU Sync Barrier Registers" on page 9). For the CPU to activate a sync barrier, it must first write to the Trigger register and then perform read polling on the virtual register.

Note

The same CPU can use the both sets of registers. in multi-CPU configurations, the two sets can be used by two CPUs (one per each CPU).

The trigger register is 3-bits wide. Each bit defines which buffers must be flushed. If for example, the sync barrier is used to flush the write path from PCI_0 to DRAM, the PCI_0 bit must be set. If it goes to a DRAM cache coherent region, the DRAM bit must be set.

A write to the CPU Sync Barrier Trigger register, triggers the sync barrier state machine. A subsequent read polling on the CPU Sync Barrier Virtual register results in value of 0xFFFF.FFFF, until the sync action completes. Once the buffers are flushed, a read of the Virtual register results in a value of 0x0.

The GT-64260B also supports nesting of sync barriers. For example, there are two NIC cards set on the two PCI interfaces. Each card has a dedicated GPP interrupt. The NIC on PCI_0 interrupts the CPU first. This indicates to the CPU that there is a pending buffer to handle in DRAM. The CPU interrupt handler triggers, in response, a sync barrier to flush the path from PCI_0 to DRAM. Then, the CPU starts polling on the CPU Sync Barrier Virtual register. However, before the sync barrier is resolved, the NIC on PCI_1 also interrupts the CPU. If the interrupt handler considers this interrupt from PCI_1 a higher priority task, the handler can re-trigger the sync barrier, setting the PCI_1 bit in the CPU Sync Barrier Trigger register. The GT-64260B sync barrier implementation guarantees that a subsequent read polling on the CPU Sync Barrier Virtual register will result in a value of 0x0, only after the path from PCI_1 to DRAM is flushed.

In multi-CPU configurations, where every CPU manages its own traffic from PCI (or Ethernet ports) to DRAM, each CPU handles its own sync barrier activity using its own set of sync barrier registers, without interfering with the activity of the peer CPU. This implementation guarantees that the two sync barriers do not mix with each other. Additionally, the CPU sync barrier is only marked as resolved (read of the Virtual register results in 0x0) when its specific path is flushed.

1.1 Basic and Enhanced Sync Barrier Usage Notes

Only one of the sync barriers or the ordering mechanisms can work at once:

- If using the basic (GT-64260A) CPU sync barrier mechanism, do not access the enhanced CPU sync barrier mechanism registers. In the Counters and Sync Barrier Attribute register, set the `SyncBarMode` bit [15] to '0' (see Table 11 on page 11).
- If working with the enhanced (GT-64260B) CPU Sync barrier mechanism, the basic CPU Configuration register's `IOSBDis` and `ConfSBDis` bits [29:28] must be set to '1', disabled. Also set the `SyncBarMode` bit [15] to '1' in the Counters and Sync Barrier Attribute register.

Note

If PCI Ordering or Deadlock modes are enabled, do not use either CPU sync barrier.

2. CPU to PCI Ordering

In many applications, the local PowerPC processor interfaces with a "protocol chip" over the PCI bus. This "protocol chip" PCI device can be a user specific FPGA, connected to user proprietary fabric, or an off the shelf PCI Application Specific Integrated Circuit (ASIC), such as an Ethernet NIC or Fiber Channel (FC) adapter.

Typically, the handshake between the local CPU and the PCI device is based on some producer-consumer model, in which buffers and descriptors are placed in the local DRAM and change ownership.

This CPU-to-PCI device handshake is based on the assumption that when the interrupt handler reads the descriptors/buffers, they are already stored in the DRAM. For this assumption to be correct, the following conditions must apply:

- The NIC only asserts an interrupt after the descriptors/packet data write over the PCI is completed. All standard PCI devices meet this requirement.
- After the interrupt handler reads NIC status, it is guaranteed that the data is stored in DRAM. In other words, the CPU read over the PCI (read of the NIC status register) must initiate the flushing of all posted write data in the system controller buffers towards the DRAM.

Note

For more details about PCI Ordering, see AN-84: *PCI Ordering Implementation* and the PCI spec.

The "traditional" sync barrier mechanism implemented in the GT-6426xA can only partially solve this problem. Enabling the CPU Configuration register's `IOSBDis` and `ConfSBDis` bits [29:28] is only a partial solution, since it activates the sync barriers only for I/O and configuration reads coming from the CPU to the PCI. Other CPU to PCI reads do not activate the sync barrier mechanism. Its use is also limited for non-cache coherent systems.

To fully meet the second condition in the GT-64260B, the correct Deadlock and Ordering register PCI BAR (where the NIC is located) ordering bit must be enabled. The PCI ordering enable bits are located in the Deadlock and Ordering register, see Table 10.

Note

To enable any of the PCI ordering bits, the Deadlock and Ordering register's PCIOrEn bit [30] must be set to '1' (see Table 10 on page 10).

When the CPU issues a read transaction targeted to one of the enabled PCI ordering addresses, the GT-64260B asserts ARTRY* on the PowerPC bus and handles it as a delayed read. This means that every new transaction to an ordering enabled address will be rejected, i.e., Artry* will be asserted by the GT-64260B.

The GT-64260B issues the read request toward the requested device and waits for its data to return. When this happens, GT-64260B issues a sync barrier toward the requested PCI (PCI0 or PCI1), and to the SDRAM unit snoop queue, if enabled. Enable the sync barrier enable bits in the Counters and Sync Barriers Attribute register, see Table 11 on page 11.

Note

If PCI ordering is enabled, the SyncBarMode bit [15] in the Counters and Sync Barriers Attributes register must be set to '1'. In the same register, set the appropriate L0SyncBar, L1SyncBar, and/or DSyncBar bits [14:12] depending on which sync barrier must be activated.

After both sync barriers are resolved, the read data is returned to the CPU, when it retries the original read address transaction. If the CPU does not retry the same address, the GT-64260B uses the Counters and Sync Barriers Attribute register's AbCnt bits [26:16] to implement a mechanism that discards the returning data after a programmable time-out.

3. Preventing Potential PCI Deadlock Mechanism

When the local PowerPC processor attached to the GT-64260A is interfacing with PCI devices, and when using cache coherency, some situations may result in a system deadlock.

Deadlocks only occur when interfacing with PCI devices that have some dependencies between their action as a master and a slave device. For instance, a device that refuses to accept a read transaction as a slave before completing a write transaction as a master.

Note

PCI simple devices, such as Ethernet NIC or FC/SCSI adapters, do not have such dependency and, therefore, are not exposed to these potential deadlocks.

For more information about deadlock situations, see AN-85: *PCI Deadlock Considerations*

The main reason that deadlocks occur, when cache coherency is enabled, is the inability of the CPU Write Back (WB) transactions to bypass the CPU interface transaction queue, when the queue is full or has an unresolved transaction toward the internal crossbar.

The new GT-64260B mechanism implements an additional queue, one to four entries in depth, capturing the WB transaction and a regret mechanism on the crossbar interface. The regret mechanism temporarily discards any "stuck" transaction(s), and allows a WB transaction to bypass any "stuck" transactions toward the SDRAM. The "stuck" transaction is retried later. Use the Write Back Priority and Buffer Depth register's RegretCnt bits [27:20] to set the regret mechanism and the WBSideBuf bits [31:30] to set the queue entry depth (see Table 12 on page 12).

Note

Currently, the RegretCnt bits [27:20] and the WBSideBuf bits [31:30] are reserved. Use the default numbers.

With the deadlock mechanism enabled, every CPU read transaction, targeted to one of the enabled deadlock addresses (PCI0 or PCI1), is retried by the GT-64260B and handled as a “delayed read”.

To program the PCI_0, PCI_1, or deadlock mechanism:

1. In the Deadlock and Ordering register, set the DLEn[31], PCI0DLEn[16] and/or PCI1DLEn[17] to ’1’ (see Table 10 on page 10).
2. Set the requested cache region enable bits in the Write Back Priority and Buffer Depth register, see Table 12 on page 12.
3. In the Counters and Sync Barriers Attributes register, set the SyncBarMode bit[15] to ’1’, see Table 11 on page 11. L0SyncBar bit[12], L1SyncBar bit[13] and DSyncBar bit[14] in the same register may be configured as requested.

Notes

- In most systems, the cache region is located only on the SDRAM addresses. Therefore, only SCS0WBEn to SCS3WBEn bits[3:0] should be programed.
- While the deadlock mechanism is enabled, a PCI cannot be configured as a cacheable region. Therefore, set bits [18:9] of the Write Back Priority and Buffer Depth register to ’0’.
- A potential deadlock situation is also possible when an agent located on one of the GT-64260B Communication unit, e.g. Ethernet port, accesses a cache regions (for more details refer to the GT-64260A errata and restrictions document). To enable the deadlock mechanism for the Communication port, set the Deadlock and Ordering register’s CommDLEn bit[18] to ’1’ and the other bits on the Write Back Priority and Buffer Depth register.
- Multi-GT mode is not fully supported when the deadlock mechanism is enabled.

4. PCI-to-CPU Ordering

The GT-64260A PCI-to-CPU Sync barrier was triggered in two ways:

1. Reading from the PCI0/PCI1 Sync Barrier Virtual Register offsets 0x1D10/0x1D90.
2. Issuing a configuration read and setting the PCIx Command register’s SBDis bit[13] to ’0’.

Note

In the GT-64260B, reading addresses that are marked as a PCI ordered access also trigger the basic sync barrier mechanism.

For the GT-64260B, a new pci_order bit [15] was added to all of the PCI1/PCI0 Access Control Base 0-7 (low) registers, at offsets 0x1E00/0x1E80 to 0x1E70/0x1EF0. Setting this bit to ’1’ enables the PCI-to-CPU ordering for that address range.

Note

If the pci_order bit is enabled, the aggressive prefetch enable bits [18:16] are ignored. This means that transactions targeted to a pci_order address are never be treated as aggressive prefetch, and only one read buffer is allocated.

5. Enhanced Aggressive Prefetch Mode

The GT-64260B provides an enhanced Aggressive Prefetch mode that provides better performance than the basic (GT-64260A) Aggressive Prefetch mode. Use this mode in systems were a single PCI master device reads long burst of data from the SDRAM.

Enable the Enhanced Aggressive prefetch mode for PCI_0 or PCI_1 by setting to ’1’ the PCIx Command register’s bit [1], at offsets 0xC00 or 0xC80.

Notes

- Bit [1] is reserved in the GT-64260A. In the GT-64260B, this bit is available for programming.
- When the Enhanced Aggressive Prefetch mode is enabled, only two of the eight read buffers are available: buffer 0 and buffer 1. The Read Buffer Discard Timer register's `RdBufEn` bits [23:16] must be set to 0x03.
- Limiting the number of usable buffers to two might cause a performance reduction in multi-PCI Master systems.

6. Transaction Flow Control and ARTRY* Assertion

The GT-64260B transaction flow control implementation is very similar to the GT-64260A.

As in the GT-64260A, the GT-64260B controls the CPU transaction rate using the AACK* signal.

When the CPU interface transaction queue is full, the GT-64260B keeps AACK* de-asserted in response to a new transaction start. AACK* is kept de-asserted until there is "room" for a new transaction in the queue.

Unlike the GT-64260A, the GT-64260B's fastest AACK* response (when the transaction queue is not full) is programmable as two or three cycles after TS* assertion, via the CPU Configuration register's AACK Delay_2 bit [25] (see Table 13 on page 15). In multi-GT configurations, the earliest GT-64260B AACK* response is two cycles after TS* assertion, regardless of the AACK Delay_2 bit.

Notes

- Unlike the GT-64260A, the GT-6426xB does not use the TADelay mode. The fastest TA* response to a write transaction is the second cycle after AACK* assertion.
- If `DLEn` or `PCIOREn` are enabled (see Table 10 on page 10), the AACK* response, to a transaction targeted to the programmed addresses, might be longer.

Additionally, several changes have occurred in the CPU Configuration register.

Since the AACK Delay and FastClk bits is no longer used in the GT-64260B, a new SingleCPU bit identifies ARTRY* sampling enable (see Table 10, "Deadlock and Ordering," on page 10).

The Single CPU mode effects only the ARTRY* sampling, in cases of data transactions. This mode has no effect on address only transactions. Enabling this mode improves the CPU to Memory latency in one cycle. In multi-CPU configurations, or single CPU configurations that might encounter a CPU self-generating ARTRY*, the SingleCPU bit must be set to '0'.

7. Multi-CPU and Symmetric Multi Processing (SMP) Support

The GT-64260B includes special features to support multi-CPU configurations. These features are summarized below.

Note

Most of the features rely on the usage of the GT-6426xB internal bus arbiter. An internal bus arbiter is required because the GT-6426xB needs the arbiter to distinguish between a CPU_0 and a CPU_1 access. If using an external bus arbiter in 60x bus mode, the GT-6426xB cannot determine if the current CPU access is driven by CPU_0 or CPU_1.

7.1 Bus Features

The internal arbiter supports a dual CPU reset sequence. After reset, CPU_1 arbitration is disabled. This means CPU_0 boots first and initializes the system. After the system is initialized, CPU_0 enables CPU_1 arbitration. See the GT-64260A datasheet's PowerPC Bus Arbitration section for full details.

After reset, CPU_1 arbitration is disabled. This means CPU_0 boots first and initializes the system. After the system is initialized, CPU_0 enables CPU_1 arbitration.

7.2 Doorbell Interrupts

The GT-6426xB has two 8-bit wide Doorbell Interrupt registers, one for each CPU. The Doorbell Interrupt register can be used for CPU-to-CPU interrupt generation, for external PCI device to CPU interrupt generation, or even for CPU to interrupt itself.

To set the interrupt, write a value of '1' to a Doorbell Interrupt register bit. A '0' value has no affect. This scheme enables distributing the eight doorbell interrupts between multiple interrupt requesters, without the interrupts interfering with one another.

There is a separate pair of Doorbell Interrupt Clear registers, one per each CPU, that are used by the CPUs to clear doorbell interrupts. A CPU writing a value of '1' to clear the register bit, clears the corresponding interrupt. A value of '0 has no affect.

Notes

- For further information, see Section 8.5 "SMP Registers" on page 17.
- Two bits were added to the Main Interrupt Cause (high) register, at offset 0xC68. Bit [12] is the CPU0_doorbell and bit [13] is CPU1_doorbell.

7.3 Who Am I Register

The GT-6426xB includes a "Who Am I" register (see Table 14 on page 17). This is a read only register, that may help the software running on the CPU, to identify itself.

A CPU_0 read from this register results in a value of 0x0. A CPU_1 read results in a value of 0x1. An external PCI agent read results in a value of 0x2.

7.4 Semaphores

The GT-6426xB supports eight semaphore registers. These registers can be used as a lock mechanism between the two CPUs, or even between a CPU and an external PCI agent.

Each possible owner has it's own ID:

- CPU0 ID = 0.
- CPU1 ID = 1.
- External agent ID = 2.

After wake up, all eight semaphores are unlocked.

The first CPU or PCI agent to read a semaphore register, receives its own ID. This ID indicates that interface owns this semaphore. Once locked by one of the three possible owners (CPU0, CPU1, PCI), a read of the semaphore register by any of the other two, will return the owner's ID. For example, if Semaphore0 is owned by CPU1, a CPU0 read will result in a value of 0x1, indicating to CPU0 that this semaphore is currently owned by CPU1.

To unlock a semaphore, the owner writes back a value of 0xFF.

7.5 Sync Barrier

The GT-6426xB supports a CPU Sync Barrier mechanism that enables handshaking between PCI agents pushing data to the GT-6426xB DRAM and the CPU reading this data. This Sync Barrier mechanism allows the CPU software to confirm that the data received from a PCI has been placed in memory.

The GT-6426xB Sync Barrier implementation enables dual CPU configuration, where both CPUs handle data received from a PCI. Each CPU can trigger its own sync barrier operation. This implementation guarantees that the two sync barrier actions do not interfere with one another.

Note

For full details on CPU sync barrier implementation, see the GT-64260A datasheet's CPU Synchronization Barrier and Section 1. "Enhanced CPU Synchronization Barriers" on page 1.

Boot Up in Multi-CPU Configuration

In a multi-CPU configuration, each CPU has its own boot code placed in a different memory location. Assuming that CPU_0 boot code is placed in BootCS# and CPU_1 code is placed in DevCS[3]#, the following boot up sequence is recommended:

1. Both CPUs assert their bus request signals. Initially, BR1# is masked and the GT-6426xB gives bus mastership only to CPU0.
2. CPU_0 reads the boot code from BootCS#. The BootCS# default address space is 0xFFC0.0000 up to FFFF.FFFF. PowerPC boot vector 0xFFF0.0100 resides within this range.
3. CPU_0 initializes the GT-6426xB internal registers, and specifically the GT-6426xB address map.
4. At some point, CPU_0 stops executing from the BootCS# and starts executing code from DRAM.
5. CPU_0 disables the BootCS# address window. Then, CPU_0 re-allocates the DevCS[3]# window to match the CPU_1 boot vector.
6. CPU_0 clears the MaskBR1 bit. This enables CPU_1 arbitration.
7. CPU_1 is now able to read its boot code from DevCS[3]#.

Note

PowerPC boot vector can be configured to 0xFFF0.0100 or 0x0000.0100. The GT-6426xB BootCS# default map matches boot vector 0xFFF0.0100. However, for CPU1, boot vector 0x0000.0100 is also useful.

8. Referenced Registers

Table 1: CPU Sync Barrier Map

Register	Offset	Page
CPU0 Sync Barrier Trigger	0xD0	page 9
CPU0 Sync Barrier Virtual	0x0C0	page 9
CPU1 Sync Barrier Trigger	0xD8	page 9
CPU1 Sync Barrier Virtual	0x0C8	page 9

Table 2: CPU to PCI Ordering Map

Register	Offset	Page
Deadlock and Ordering	0x2D0	page 10
Counters and Sync Barrier Attribute	0x2E0	page 11

Table 3: Write Back Transaction Map

Register	Offset	Page
Write Back Priority and Buffer Depth	0x2D8	page 12

Table 4: CPU Configuration Map

Register	Offset	Page
CPU Configuration	0x000	page 15

Table 5: SMP Register Map

Register	Offset	Page
Who Am I	0x200	page 17
CPU0 Doorbell	0x214	page 17
CPU0 Doorbell Clear	0x21C	page 18
CPU1 Doorbell	0x224	page 18
CPU1 Doorbell Clear	0x22C	page 18
CPU0 Doorbell Mask	0x234	page 18
CPU1 Doorbell Mask	0x23C	page 19
Semaphore0	0x244	page 19
Semaphore1	0x24C	page 19
Semaphore2	0x254	page 19
Semaphore3	0x25C	page 19
Semaphore4	0x264	page 20
Semaphore5	0x26C	page 20
Semaphore6	0x274	page 20
Semaphore7	0x27C	page 20

8.1 CPU Sync Barrier Registers

Table 6: CPU0 Sync Barrier Trigger
Offset: 0xD0

Bits	Field	Type/ Init Val	Description
2:0	SBTrig	RW 0x0	Sync Barrier Trigger. A write to this register triggers the sync barrier process. The three bits, define which buffers should be flushed. 0 = PCI_0 slave write buffer 1 = PCI_1 slave write buffer 2 = SDRAM snoop queue
31:3	Reserved	RES 0x0	Reserved.

Table 7: CPU0 Sync Barrier Virtual
Offset: 0x0C0

Bits	Field	Type/ Init Val	Description
31:0	SBStat	RW 0x0	As long as sync barrier is in progress, a read from this register results in a value of 0xFFFF.FFFF. As soon as sync barrier is resolved, a read results in 0x0.

Table 8: CPU1 Sync Barrier Trigger
Offset: 0xD8

Bits	Field	Type/ Init Val	Description
31:0	SBTrig	RW 0x0	The same as the CPU0 Sync Barrier Trigger register

Table 9: CPU1 Sync Barrier Virtual
Offset: 0x0C8

Bits	Field	Type/ Init Val	Description
31:0	SBStat	RW 0x0	The same as the CPU0 Sync Barrier Virtual register

8.2 CPU to PCI Ordering Registers

Table 10: Deadlock and Ordering
Offset: 0x2D0

Bits	Field	Type/ Init Val	Description
0	PCI0OrEn	RW 0x0	PCI_0 In/Out Order Enable This bit activates the sync barrier ordering when approaching this BAR. It is masked if PCIOrEn bit [30] is 0x0 (disabled)
1	PCI0Mem0OrEn	RW 0x0	The same as PCI0OrEn.
2	PCI0Mem1OrEn	RW 0x0	The same as PCI0OrEn.
3	PCI0Mem2OrEn	RW 0x0	The same as PCI0OrEn.
4	PCI0Mem3OrEn	RW 0x0	The same as PCI0OrEn.
5	PCI1OrEn	RW 0x0	The same as PCI0OrEn.
6	PCI1Mem0OrEn	RW 0x0	The same as PCI0OrEn.
7	PCI1Mem1OrEn	RW 0x0	The same as PCI0OrEn.
8	PCI1Mem2OrEn	RW 0x0	The same as PCI0OrEn.
9	PCI1Mem3OrEn	RW 0x0	The same as PCI0OrEn.
15:10	Reserved	RES 0x0	Reserved.
16	PCI0DLEn	RW 0x0	PCI_0 Deadlock Enable This bit activates the PCI_0 deadlock mechanism. 0 = Disable 1 = Enable
17	PCI1DLEn	RW 0x0	PCI_1 Deadlock Enable This bit activates the PCI_1 deadlock mechanism. 0 = Disable 1 = Enable
18	CommDLEn	RW 0x0	Communication Unit Deadlock Enable This bit activates the communication unit's deadlock mechanism. 0 = Disable 1 = Enable
28:19	Reserved	RES 0x0	Reserved.

Table 10: Deadlock and Ordering (Continued)
Offset: 0x2D0

Bits	Field	Type/ Init Val	Description
29	SingleCPU	RW 0x0	Single CPU This bit sets the configuration to work with a single CPU or with multi-CPUs. 0 = Multi-CPUs 1 = Single CPU This bit must be cleared in multi-CPU or single CPU configuration that might face a CPU self-generating ARTRY* signal.
30	PCIOrEn	RW 0x0	PCI Ordering Mechanism Enable This bit activates the PCI ordering mechanism. 0 = Disable 1 = Enable
31	DLEn	RW 0x0	Deadlock Mechanism Enable This bit enables the deadlock mechanism. 0 = Disable 1 = Enable

Table 11: Counters and Sync Barrier Attribute
Offset: 0x2E0

Bits	Field	Type/ Init Val	Description
7:0	NoMatchExtraCnt	RW 0x0	CPU Address No Match Extended Counter
11:8	AackDlyCnt	RW 0x6	Address Acknowledge Delay Counter **NOTE:** For Marvell use only.
12	L0SyncBar	RW 0x0	This bit enables the sync barrier mechanism towards PCI_0. 0 = Disable 1 = Enable **NOTE:** This bit is masked if `PCIOrEn` and `DLEn` are disabled, see Table 10 on page 10.
13	L1SyncBar	RW 0x0	This bit enables the sync barrier mechanism towards PCI_1. 0 = Disable 1 = Enable **NOTE:** Masked if PCIOrEn and DLEn are disabled, see Table 10 on page 10.
14	DSyncBar	RW 0x0	This bit enables the sync barrier mechanism towards the SDRAM or the devices. 0 = Disable 1 = Enable **NOTE:** This bit is masked if `PCIOrEn` and `DLEn` are disabled, see Table 10 on page 10.

Table 11: Counters and Sync Barrier Attribute (Continued)
Offset: 0x2E0

Bits	Field	Type/ Init Val	Description
15	SyncBarMode	RW 0x0	This bit enables the enhanced sync barrier mechanism, see Section 1. "Enhanced CPU Synchronization Barriers" on page 1. 0 = Standard sync barrier mechanism (GT-64260A) 1 = Enhanced sync barrier mechanism (GT-64260B) **NOTE:** If disabled, see the CPU Synchronization Barrier section in the GT-64260A datasheet for a description of the CPU sync barrier mechanism.
26:16	AbCnt	RW 0x07FF	Abort Counter Expiration of this counter aborts the ordering "return data mechanism." **NOTE:** For Marvell use only.
31:27	Reserved	RES 0x0	Reserved.

8.3 Write Back Transaction Register

Table 12: Write Back Priority and Buffer Depth
Offset: 0x2D8

Bits	Field	Type/ Init Val	Description
0	SCS0WBEn	RW 0x0	SCS_0 Write Back Enable A WB transaction to SCS_0 goes to the side buffer queue. 0 = Disable 1 = Enable
1	SCS1WBEn	RW 0x0	SCS_1 Write Back Enable A WB transaction to SCS_1 goes to the side buffer queue. 0 = Disable 1 = Enable
2	SCS2WBEn	RW 0x0	SCS_2 Write Back Enable A WB transaction to SCS_2 goes to the side buffer queue. 0 = Disable 1 = Enable
3	SCS3WBEn	RW 0x0	SCS_3 Write Back Enable A WB transaction to SCS_3 goes to the side buffer queue. 0 = Disable 1 = Enable
4	CS0WBEn	RW 0x0	CS_0 Write Back Enable A WB transaction to CS_0 goes to the side buffer queue. 0 = Disable 1 = Enable

Table 12: Write Back Priority and Buffer Depth (Continued)
Offset: 0x2D8

Bits	Field	Type/ Init Val	Description
5	CS1WBEn	RW 0x0	CS_1 Write Back Enable A WB transaction to CS_1 goes to the side buffer queue. 0 = Disable 1 = Enable
6	CS2WBEn	RW 0x0	CS_2 Write Back Enable A WB transaction to CS_2 goes to the side buffer queue. 0 = Disable 1 = Enable
7	CS3WBEn	RW 0x0	CS_3 Write Back Enable A WB transaction to CS_3 goes to the side buffer queue. 0 = Disable 1 = Enable
8	BootCSWBEn	RW 0x0	Boot CS Write Back Enable A WB transaction to the boot CS goes to the side buffer queue. 0 = Disable 1 = Enable
9	PCI0IOWBEn	RW 0x0	PCI_0 I/O Write Back Enable A WB transaction to PCI_0 I/O goes to the side buffer queue. 0 = Disable 1 = Enable
10	PCI0Mem0WBEn	RW 0x0	PCI_0 Memory_0 Write Back Enable A WB transaction to PCI_0 Mem_0 goes to the side buffer queue. 0 = Disable 1 = Enable
11	PCI0Mem1WBEn	RW 0x0	PCI_0 Memory_1 Write Back Enable A WB transaction to PCI_0 Mem_1 goes to the side buffer queue. 0 = Disable 1 = Enable
12	PCI0Mem2WBEn	RW 0x0	PCI_0 Memory_2 Write Back Enable A WB transaction to PCI_0 Mem_2 goes to the side buffer queue. 0 = Disable 1 = Enable
13	PCI0Mem3WBEn	RW 0x0	PCI_0 Memory_3 Write Back Enable A WB transaction to PCI_0 Mem_3 goes to the side buffer queue. 0 = Disable 1 = Enable
14	PCI1IOWBEn	RW 0x0	PCI_1 I/O Write Back Enable A WB transaction to PCI_0 I/O goes to the side buffer queue. 0 = Disable 1 = Enable

Table 12: Write Back Priority and Buffer Depth (Continued)
Offset: 0x2D8

Bits	Field	Type/ Init Val	Description
15	PCI1Mem0WBEn	RW 0x0	PCI_1 Memory_0 Write Back Enable A WB transaction to PCI_0 Mem_0 goes to the side buffer queue. 0 = Disable 1 = Enable
16	PCI1Mem1WBEn	RW 0x0	PCI_1 Memory_1 Write Back Enable A WB transaction to PCI_0 Mem_1 goes to the side buffer queue. 0 = Disable 1 = Enable
17	PCI1Mem2WBEn	RW 0x0	PCI_1 Memory_2 Write Back Enable A WB transaction to PCI_0 Mem_2 goes to the side buffer queue. 0 = Disable 1 = Enable
18	PCI1Mem3WBEn	RW 0x0	PCI_1 Memory_3 Write Back Enable A WB transaction to PCI_0 Mem_3 goes to the side buffer queue. 0 = Disable 1 = Enable
19	Reserved	RES 0x0	Reserved.
27:20	RegretCnt	RW 0x40	PCI Regret Counter These bits set the PCI timer for transactions going to the crossbar. **NOTE:** For Marvell use only.
29:28	Reserved	RES 0x0	Reserved.
31:30	WBSideBuf	RW 0x3	Write Back Side Buffer Depth These bits indicate the number of WB transactions held as entries in this queue. 0x0 = 1 entry 0x1 = 2 entries 0x2 = 3 entries 0x3 = 4 entries **NOTE:** For Marvell use only.

8.4 CPU Configuration Register

Table 13: CPU Configuration
Offset: 0x000

NOTE: The bits in this register are for the GT-6426xB implementation.The changes from the GT-64260A implementation are noted.

Bits	Field	Type/ Init Val	Description
7:0	NoMatchCnt	RW 0xFF	CPU Address Miss Counter **NOTE:** An extension of this counter is available in Table 11, "Counters and Sync Barrier Attribute," on page 11.
8	NoMatchCntEn	RW 0xFF	CPU Address Miss Counter Enable This bit is relevant only if multi-GT is enabled. 0 = Disabled 1 = Enabled.
9	NoMatchCntExt	RW 0xFF	CPU address miss counter MSB **NOTE:** An extension of this counter is available Table 11, "Counters and Sync Barrier Attribute," on page 11.
10	Reserved	RES 0x0	Reserved.
11	Reserved	RES 0x1	Reserved. **NOTE:** In the GT-6426xB this is the AACK Delay bit.
12	Endianess	AD[4] sampled at reset.	Must be 0 **NOTE:** The GT-6426xB does not support the PowerPC Little Endian convention.
13	Pipeline	RW 0x0	Pipeline Enable 0 = Disabled. The GT-6426xB will not respond with AACK* to a new CPU transaction, before the previous transaction data phase completes. 1 = Enabled
14	Reserved	RES 0x0	Reserved.
15	Reserved	RES 0x1	Reserved. **NOTE:** In the GT-64260A, this is the TADelay bit.
16	RdOOO	RW 0x0	Read Out of Order Completion 0 = Not Supported Data is always returned in order (DTI[0-2] is always driven 0x0). 1 = Supported
17	Stop Retry	RW 0x0	This bit is relevant only if PCI Retry is enabled 0 = Keep retrying all PCI transactions targeted to the GT-6426xB. 1 = Stop retrying all PCI transactions.
18	MultiGTDec	Reset Initialization	Multi-GT Address Decode 0 = Normal address decoding 1 = Multi-GT address decoding

Table 13: CPU Configuration
Offset: 0x000 (Continued)

NOTE: The bits in this register are for the GT-6426xB implementation.The changes from the GT-64260A implementation are noted.

Bits	Field	Type/ Init Val	Description
19	DPValid	RW 0x0	CPU DP[0-7] Connection 0 = Not connected. CPU write parity is not checked. 1 = Connected
21:20	Reserved	RES 0x0	Reserved.
22	PErrProp	RW 0x0	Parity Error Propagation 0 = GT-6426xB always drives good parity on DP[0-7] during CPU reads. 1 = GT-6426xB drives bad parity on DP[0-7] in case the read response from the target interface comes with an erroneous data indication (e.g., ECC error from SDRAM interface).
23	Reserved	RES 0x1	Reserved. **NOTE:** In the GT-64260A, this is the FastClk bit.
24	Reserved	RES 0x0	Reserved.
25	AACK Delay_2	RW 0x1	Address Acknowledge Delay 2 0 = Earliest assertion of AACK* is two cycles after TS* assertion. 1 = Earliest assertion of AACK* is three cycles after TS* assertion. **NOTE:** This bit is useful for interfacing CPUs when their ARTRY* window is delayed. This bit is not supported in multi-GT mode. If multi-GT is enabled, it is impossible to interface CPUs in which the ARTRY* window is delayed. For example: If a system is using MPC7450 CPU, the CPU core clock ratio must be selected to be higher than or equal to 1:5.
26	APValid	RW 0x0	CPU AP[0-3] Connection 0 = Not connected. The CPU address parity is not checked. 1 = Connected
27	RemapWrDis	RW 0x0	Address Remap Registers Write Control 0 = Write to Low Address decode register. This setting results in writing of the corresponding Remap register. 1 = Write to Low Address decode register. This setting has no affect on the corresponding Remap register.
28	ConfSBDis	RW 0x1	Configuration Read Sync Barrier Disable 0 = Sync Barrier enabled 1 = Sync Barrier disabled **NOTE:** These configuration bits are part of the ordering mechanism. For more details, see Section 2. "CPU to PCI Ordering" on page 2.

Table 13: CPU Configuration
Offset: 0x000 (Continued)

NOTE: The bits in this register are for the GT-6426xB implementation.The changes from the GT-64260A implementation are noted.

Bits	Field	Type/ Init Val	Description
29	IOSBDis	RW 0x1	I/O Read Sync Barrier Disable 0 = Sync Barrier enabled 1 = Sync Barrier disabled **NOTE:** These configuration bits are part of the ordering mechanism. For more details, see Section 2. "CPU to PCI Ordering" on page 2.
30	ClkSync	AD[5] sampled at reset.	Clocks Synchronization 0 = The CPU interface is running with the SysClk, which is asynchronous to TClk. 1 = The CPU interface is running with the TClk.
31	Reserved	RES 0x0	Reserved.

8.5 SMP Registers

Note

For proper operation of the following registers, the GT-6426xB CPU bus arbiter must be used.

Table 14: Who Am I
Offset: 0x200

Bits	Field	Type/ Init Val	Description
1:0	ID	0x0	Read Only virtual register. If read by CPU0, it returns 0x0. If read by CPU1, it return 0x1. If read by the PCI agent, it returns 0x2.
31:2	Reserved	0x0	Reserved.

Table 15: CPU0 Doorbell
Offset: 0x214

Bits	Field	Type/ Init Val	Description
7:0	Doorbell	0x0	This bit is set to ‘1’ upon a write value of ‘1’. Write 0 has no affect. **NOTE:** This field is cleared only by writing to the CPU0 Doorbell Clear register.
31:8	Reserved	0x0	Reserved.

Table 16: CPU0 Doorbell Clear
Offset: 0x21C

Bits	Field	Type/ Init Val	Description
7:0	Clear	0x0	Virtual register. CPU0 write value of '1' clears the corresponding bit in CPU0 Doorbell register. A write of '0' has no affect. A write from CPU1 or from some PCI agent has no affect. A read returns the CPU0 Doorbell register value.
31:8	Reserved	0x0	Reserved.

Table 17: CPU1 Doorbell
Offset: 0x224

Bits	Field	Type/ Init Val	Description
7:0	Doorbell	0x0	This bit is set to '1' upon a write value of '1'. A write of '0' has no affect. **NOTE:** This bit is cleared only by writing to the CPU1 Doorbell Clear register.
31:8	Reserved	0x0	Reserved.

Table 18: CPU1 Doorbell Clear
Offset: 0x22C

Bits	Field	Type/ Init Val	Description
7:0	Clear	0x0	Virtual register. CPU1 write value of '1' clears the corresponding bit in CPU1 Doorbell register. A write of 0 has no affect. A write from CPU0 or from some PCI agent has no affect. A read returns the CPU1 Doorbell register value.
31:8	Reserved	0x0	Reserved.

Table 19: CPU0 Doorbell Mask
Offset: 0x234

Bits	Field	Type/ Init Val	Description
7:0	Mask	0x0	Mask bit per doorbell cause bit. 0 = The corresponding doorbell interrupt is masked. 1 = Enabled.
31:8	Reserved	0x0	Reserved.

Table 20: CPU1 Doorbell Mask
Offset: 0x23C

Bits	Field	Type/ Init Val	Description
7:0	Mask	0x0	These bits are the same as in the CPU0 Doorbell Mask register.
31:8	Reserved	0x0	Reserved.

Table 21: Semaphore0
Offset: 0x244

Bits	Field	Type/ Init Val	Description
7:0	Semaphore	0x0	Read only. The semaphore can be locked by CPU0, CPU1 or the PCI agent. The first one to read the field before it is locked, receives its ID value: 0x0 for CPU0, 0x1 for CPU1, 0x2 for PCI agent. Once the field is locked by one of the three agents, a read by any of the other two will return the owner ID. Unlock the semaphore by writing back a value of 0xFF. A write of any other value is ignored.
31:8	Reserved	0x0	Reserved.

Table 22: Semaphore1
Offset: 0x24C

Bits	Field	Type/ Init Val	Description
7:0	Semaphore	0x0	These bits are the same as Semaphore0.
31:8	Reserved	0x0	Reserved.

Table 23: Semaphore2
Offset: 0x254

Bits	Field	Type/ Init Val	Description
7:0	Semaphore	0x0	These bits are the same as Semaphore0.
31:8	Reserved	0x0	Reserved.

Table 24: Semaphore3
Offset: 0x25C

Bits	Field	Type/ Init Val	Description
7:0	Semaphore	0x0	These bits are the same as Semaphore0.
31:8	Reserved	0x0	Reserved.

Table 25: Semaphore4
Offset: 0x264

Bits	Field	Type/ Init Val	Description
7:0	Semaphore	0x0	These bits are the same as Semaphore0.
31:8	Reserved	0x0	Reserved.

Table 26: Semaphore5
Offset: 0x26C

Bits	Field	Type/ Init Val	Description
7:0	Semaphore	0x0	These bits are the same as Semaphore0.
31:8	Reserved	0x0	Reserved.

Table 27: Semaphore6
Offset: 0x274

Bits	Field	Type/ Init Val	Description
7:0	Semaphore	0x0	These bits are the same as Semaphore0.
31:8	Reserved	0x0	Reserved.

Table 28: Semaphore7
Offset: 0x27C

Bits	Field	Type/ Init Val	Description
7:0	Semaphore	0x0	These bits are the same as Semaphore0.
31:8	Reserved	0x0	Reserved.

Marvell
700 First Avenue
Sunnyvale, CA 94089
Phone: (408)222 2500
Sales Fax: (408)752 9029
Email commsales@marvell.com