Universe (CA91C042) Errata

603 March Rd., Kanata, Ontario, K2K 2M5
Tel: (613) 592-0714 • 1-800-267-7231 • Fax: (613) 592-1320
Visit our Website at http://www.tundra.com

Document Number: 8091042.ER601.02

Universe (CA91C042) Errata

The current Universe design is undergoing a revision to produce the Universe II (CA91C142). The Universe II will be pin- and function-compatible with the previous version, ensuring that it can be used in existing Universe sockets. No pricing changes are currently expected between the Universe and Universe II revisions.

During this revision, Tundra is fixing any outstanding errata in the device, as well as enhancing functionality. The table below lists the errata that are addressed in this document.

A short form Universe II datasheet is currently available for download from our website at www.tundra.com. Please contact Tundra if you have any questions regarding the upcoming Universe II release.

Universe Device Errata

Errata #	Description	Fixed in Universe II Design
1	Local Slave Channel decoupled bus error handling	Yes
2	VME requester fair mode failure	Yes
3	PCI target fails to check AD[1:0] on memory writes	Yes
4	VME BGxOUT#[] lines handled improperly during SYSRST#	Yes
5	Register space consumes 64K instead of 4K	Yes
6	Local Master MAXRTRY failure	Yes
7	Noise sensitivity on VDTACK#	Yes
8	Local Slave Channel extra posted write	Yes
9	DMA Channel decoupled bus error handling	Yes
10	RMW failure; VME read uses incorrect byte lane mapping	Yes
11	PCI reset and PCI Slave Image 0	Yes
12	Local Slave Channel Disabled Byte Lane Cycle	Yes
13	Channel Deadlock - Superseded by errata #19	Yes
14	DMA Write Cycles with PCI Aligned Burst Size = 64 Bytes	Yes
15	VME Interrupt handling	Yes
16	BGnt glitches	Yes
17	Coupled Window Timer and Release-On-Request	Yes
18	LSC’s Coupled Request Timer	Yes
19	VME Master operation after VME Slave FIFO full	Yes

Universe Manual Errata

Errata #	Description	Fixed in Universe Manual Addendum
1	Figures 4.1 and 4.2 labeled incorrectly	Yes
2	Incorrect ACFAIL and SYSFAIL signal type specifications	Yes
3	Use of REQ64#	Yes

I UNIVERSE DEVICE ERRATA

1. LOCAL SLAVE DECOUPLED CYCLES

Description

The PCI Slave Channel FIFO is designed to purge the remaining entries of the offending transaction when a VME bus error (BERR*) occurs. When a VME bus error occurs due to a transaction that contains a single data beat, the FIFO will purge the single data beat which caused the bus error as well as the next transaction in the FIFO.

Impact

The second transaction will be purged needlessly, and the Universe will continue to assert BBSY# after the VME bus error occurs. If the second transaction happens to be the last decoupled transaction, the VME Master will hold BBSY# asserted, and thus hold ownership of the VME bus. The decoupled channel will remain the owner of the VME master, hence all PCI Slave Channel coupled accesses will be retried. If the unnecessarily purged transaction (the second transaction) is not the last decoupled transaction, the PCI Slave Channel will accept other decoupled transactions and process them correctly, releasing the bus appropriately when the FIFO empties.

Solution

If a VME bus error occurs as a result of a PCI Slave Channel FIFO transaction, send another transaction through the FIFO to clear the BBSY# signal asserted by the VME bus master. Read the Universe Error Logs to determine the address of the transaction that created the VME bus error; then try to determine whether a single data beat transaction created the bus error. If a single data beat transaction created the error, that implies that the second transaction has been purged, and that appropriate action is to be taken.

2. VME MASTER FAIR MODE FAILURE

Description

When the Universe is set in the FAIR request mode, it fails to acknowledge the BGx* after it has asserted its BRx#.

Impact

When the Universe fails to acknowledge a BGx* awarded to it, and there are no other VME Masters requesting the VME bus at the same request level, an arbitration time-out occurs. This results in lost VME bandwidth. If another VME Master requests the VME bus at the same request level, the Universe will surrender the BGx* to the other masters.

Solution

It is recommended that the FAIR request mode not be used in the Universe. The DEMAND request mode should always be used.

3. PCI MEMORY WRITES CYCLE MAPPING

Description

PCI Specification 2.1 requires that **memory** writes to a PCI target that have AD[1:0] = 2’b01 or 2’b1x during their address phases be disconnected after the first data transfer. However, the Universe ignores the state of AD[1:0] during the address phase.

Impact

The Universe will process all **memory** write cycles in the Linear Incrementing mode (AD[1:0] = 2b00).

Solution

Avoid using modes other than the linear incrementing mode when accessing the Universe.

4. BGXOUT# STATE DURING SYSTEM RESET

Description

While the Universe is in system reset, it will directly propagate the values on its BGxIN# inputs to its BGxOUT# outputs. In most situations this will not cause any problems since the BGxIN# inputs are at a stable high value during reset. In slot 1, however, this is not the case. The inputs are not typically at stable high values, and in fact BG3IN# with its internal pull-down resistor is in a low state. At a minimum, this will result in a low state on the Universe's BG3OUT# during reset.

The BG3 daisy chain is used during reset by some VME interfaces, including Tundra’s SCV64, for auto-syscon functionality. When the BG3IN# input is low, the VME interface will become the system controller.

Because the Universe in slot 1 will be propagating a low on BG3OUT#, a card in slot 2 with auto-syscon functionality may, along with the slot 1 Universe, become system controller. Problems may also be introduced if the slot 2 card is relying on stable high values on BG[2:0]IN*.

Impact

If the Universe is in slot 1, and the card in the next slot contains Auto System Controller functionality, its system controller functions will be enabled. The VME chassis will then have two cards as SYSCON and they will both try to arbitrate the VME bus with possible conflicts on the SYSCLK and BCLR* signals.

Solution

There exist four possible solutions:

(1) Figure 1.1 is a circuit that can be built to mask the state of the Universe BGxOUT# signals during system reset. The circuit shows an implementation with BG3OUT# only. While SYSRST# is asserted, the inversion of the VRSYSRST# after the delay element, will mask the state of BG3OUT# during and for 150ns after SYSRST* assertion. The delay element is required since BG3OUT# may remain low for several CLK64 periods after release from reset. This can be implemented in any fashion that will delay the effect of SYSRST* negation for 150ns or greater. For other bus grant daisy chain signals, the circuit may be duplicated or 10k pull-ups may be placed on each of the BG[2:0]IN# to ensure the bus grant output remains high during reset.

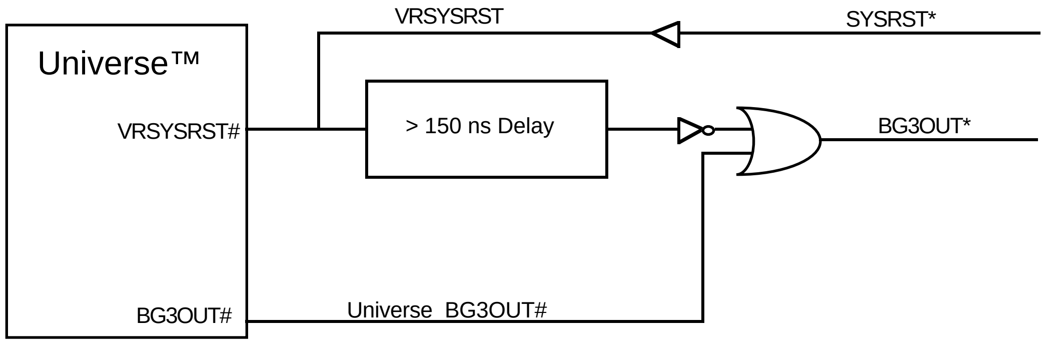

Figure 1.1: BG3OUT# Circuit Solution

(2) Place in slot 2 a card which does not have auto-syscon capability. It will not propagate the low on the bus grant daisy chain to other cards, and will not become system controller.

(3) Power-up the VME Chassis without modifications to the Universe. Through software, disable any other boards that come up as SYSCON.

(4) Use a 2.2kΩ resistor to pull-up BG3IN#. This will prevent the Universe from automatically becoming system controller. Then through software, set the Universe that is in slot 1 as system controller by setting the SYSCON bit in the MISC_CTL register. SYSCON determination on this board will have to be through a jumper or dip-switch setting read by the processor sometime after reset.

5. UNIVERSE REGISTER SPACE

Description

The Universe register space is defined as a 4K space, whose base address can be defined by programming the PCI_BS register or the VRAI_BS register. However, the actual size of the Universe register space is 64K.

Impact

64K of PCI space is consumed instead of 4K.

Solution

The user should be careful not to map a valid slave image within the 64K Universe address space; otherwise, the registers may be overwritten.

6. LOCAL MASTER MAXIMUM RETRY COUNTER

Description

The maximum number of retries before the Universe PCI master interface signals an error condition is defined by the MAXRTRY field in the MAST_CTL register (Table A.56). The retry counter fails to be reset by successfully completed, decoupled, single beat transactions. This means that the next transaction will start with the retry counter initialized to a lower value. Eventually, after many transactions, the retry counter could decrement to a point where the Universe will consider the transaction timed out after only a couple of retries on the PCI bus.

Impact

Decoupled transactions from the RXFIFO could be dropped by the Universe before they reach the destination. When this happens the Universe uses the L_CMDERR register (Table A.31) to log the command information for the transaction (CMDERR [3:0]). The address of the errored transaction is latched in the L_AERR register (Table A.32). An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the LERR interrupt is enabled (see "The Interrupt Channel" in section 2.6). If the transaction was coupled, a BERR* on the VME side would occur without any interrupt or error logging.

Solution

The counter should be disabled upon initialization of the device by programming the MAXRTRY bits to 0000 in the MAST_CTL register (Table A.56). This will allow an infinite number of retries on the PCI bus. The retry counter is not a required feature of PCI, thus disabling it will not affect PCI compliance. The default setting for the counter is enabled.

7. DTACK* NOISE

Description

The low to high transition on DTACK* is inherently noisy because most slaves rely on the backplane termination to passively pull the signal high (i.e. they do not actively drive DTACK* high). Noise on the rising edge of the DTACK* signal in the area of the threshold region can be propagated by the transceiver logic to the Universe such that the Universe recognizes a second falling edge on DTACK. This second edge on DTACK will cause the VME interface on the Universe to lock-up. The amount of noise required to cause the problem will depend upon the frequency and magnitude of the noise. Using F125 transceivers, a pulse width of 6 to 9ns has been observed to propagate through the transceiver and cause the lock-up condition.

Impact

1. When the Universe recognizes an incorrect, second falling edge on DTACK* during a BLT or MBLT cycle, it will assert AS* and BBSY* forever, or

2. The Universe may "Drop" data beats during a BLT or MBLT transfer.

Solution

Most slaves release DTACK* and rely on the termination resistors to pull the signal high, thus this is the only edge that needs to be filtered. Possible implementations of noise filtering between the transceiver and the Universe DTACK# pin include any combination of synchronizers, RC filters and Schmitt triggers.

8. POSTED WRITE BURSTS TO PCI SLAVE CHANNEL

Description

The Universe is designed to accept posted write bursts, from a PCI master, and transfer the data to the VMEbus. Under some conditions, when operating on a 32-bit PCI bus, the Universe may insert an extraneous data beat on the VME side. This will occur under the following conditions:

- operation on a 32-bit PCI bus
- posted (decoupled) write operations
- multiple data beat PCI transactions
- the last data beat of the burst has an address in which A2=0

The problem arises independent of the enabling of byte lanes during the PCI transaction, and independent of the settings of the accessed PCI slave image. Single data beat PCI transactions are not affected.

When the above conditions are met, the Universe will dequeue the data from the FIFO onto the VME bus appropriately. However, an extraneous VMEbus write cycle will be generated. This extra write will be at an address 8 bytes higher than the second last data beat, and will contain data equal to that in the second last data beat. As with all PCI to VME transactions, if the accessed PCI slave image was set for a data width less than 32 bits, multiple VME cycles will be generated depending upon the number of byte lanes enabled for the second last data beat of the affected PCI transaction. Please refer to Figures 1.2-1.4 for an illustration of how this problem is manifested.

Impact

When all conditions for this errata are met, extraneous data will be written to the VME bus at an address beyond the block of data being written. If the data is being written to unused VME memory, then the user will perceive no impact. In other situations, valid data will be overwritten by this operation. If the extraneous data is written to the control register of some device, the state of the system could be altered.

Solution

(1) Ensure that all burst transactions on PCI to the Universe are 64-bit aligned to ensure that the last data beat of the burst does not have an address in which A2=0.
(2) Since the problem only arises during decoupled operation, avoid posted writes. This may be done on a per PCI slave image basis, allowing the usage of the faster decoupled operation in some slave images for transactions that will not encounter the problem, or where its impact is not of concern.
(3) Use single beat transactions only. The error only occurs on multiple data beat bursts through the local PCI slave channel. This has the potential impact of lower overall performance and increased PCI bus bandwidth utilization.
(4) Use the Universe DMA engine to move the data from PCI to VME.
(5) Use 64-bit PCI operation whenever possible.

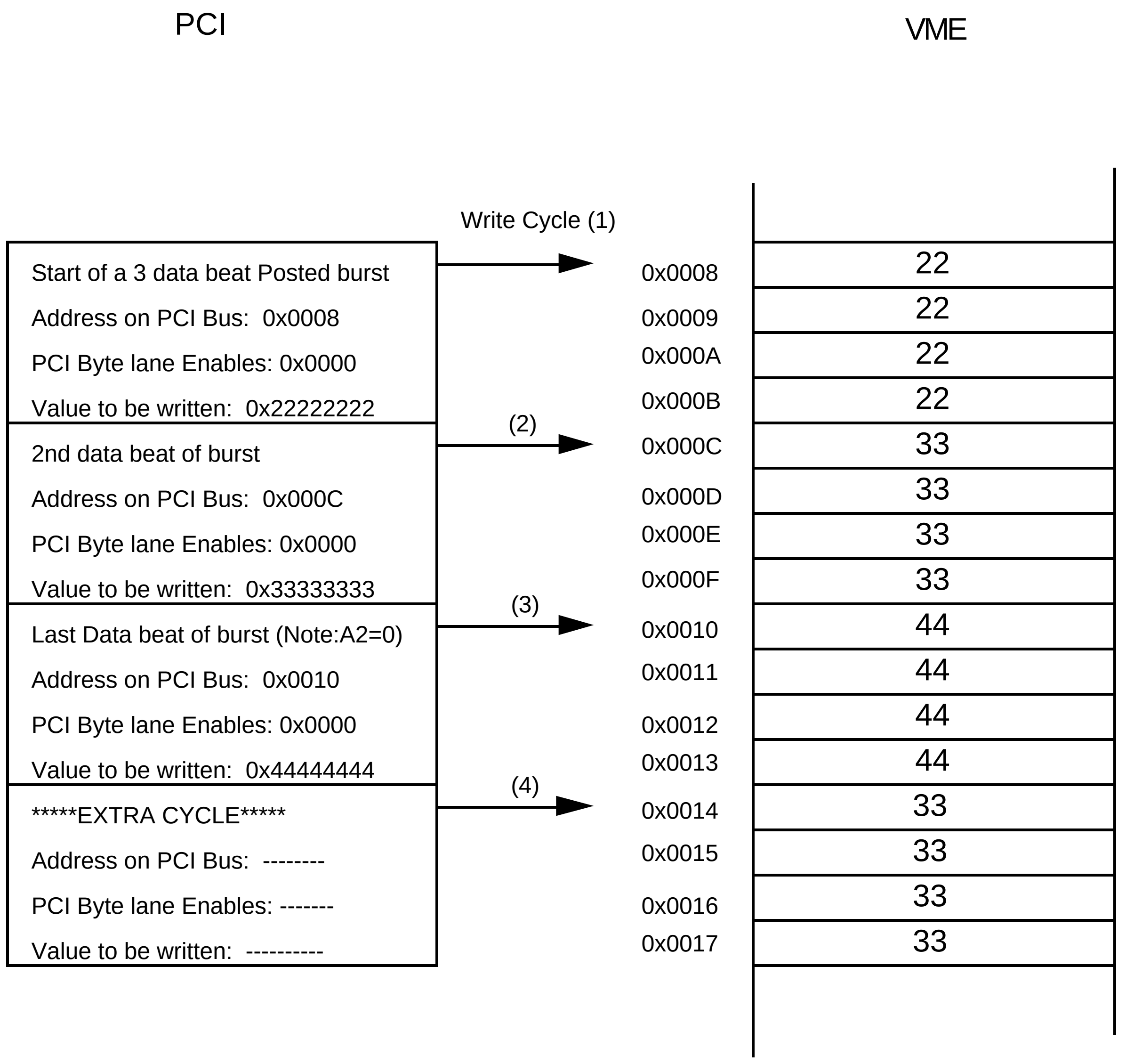
PCI
VME
Write Cycle (1)
(2)
(3)
(4)
Start of a 3 data beat Posted burst
Address on PCI Bus: 0x0008
PCI Byte lane Enables: 0x0000
Value to be written: 0x22222222
2nd data beat of burst
Address on PCI Bus: 0x000C
PCI Byte lane Enables: 0x0000
Value to be written: 0x33333333
Last Data beat of burst (Note:A2=0)
Address on PCI Bus: 0x0010
PCI Byte lane Enables: 0x0000
Value to be written: 0x44444444
*****EXTRA CYCLE*****
Address on PCI Bus: --------
PCI Byte lane Enables: -------
Value to be written: ----------
0x0008 22
0x0009 22
0x000A 22
0x000B 22
0x000C 33
0x000D 33
0x000E 33
0x000F 33
0x0010 44
0x0011 44
0x0012 44
0x0013 44
0x0014 33
0x0015 33
0x0016 33
0x0017 33

Figure 1.2: All Byte lanes enabled ---D32
(Zero translation offset; VME Maximum data width is 32 bits)

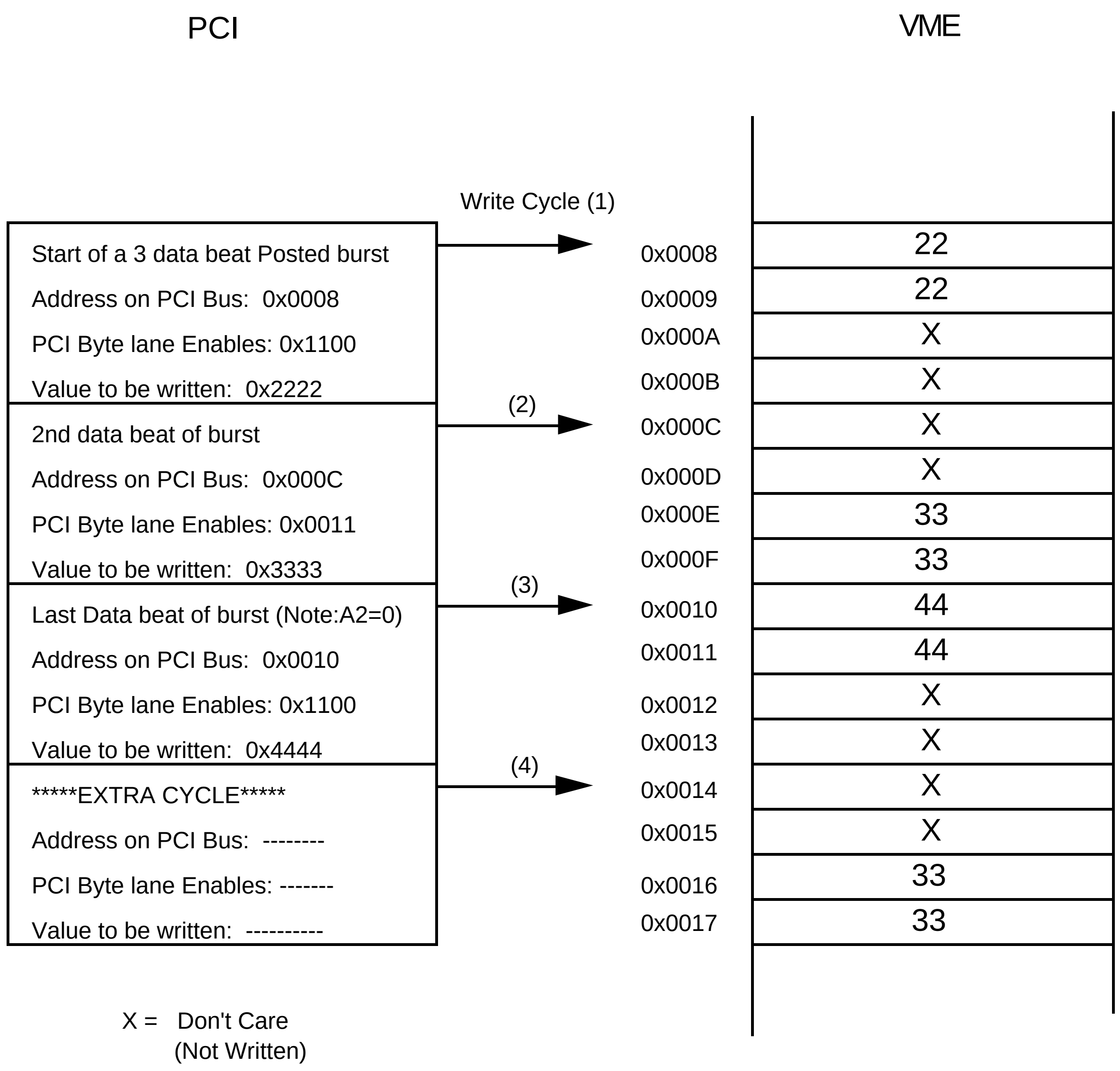
PCI
VME
Write Cycle (1)
Start of a 3 data beat Posted burst
Address on PCI Bus: 0x0008
PCI Byte lane Enables: 0x1100
Value to be written: 0x2222
(2)
2nd data beat of burst
Address on PCI Bus: 0x000C
PCI Byte lane Enables: 0x0011
Value to be written: 0x3333
(3)
Last Data beat of burst (Note:A2=0)
Address on PCI Bus: 0x0010
PCI Byte lane Enables: 0x1100
Value to be written: 0x4444
(4)
*****EXTRA CYCLE*****
Address on PCI Bus: --------
PCI Byte lane Enables: -------
Value to be written: ----------
0x0008 22
0x0009 22
0x000A X
0x000B X
0x000C X
0x000D X
0x000E 33
0x000F 33
0x0010 44
0x0011 44
0x0012 X
0x0013 X
0x0014 X
0x0015 X
0x0016 33
0x0017 33
X = Don't Care (Not Written)

Figure 1.3: Two Byte lanes enabled ---D16
(Zero translation offset; VME Maximum data width is 32 bits)

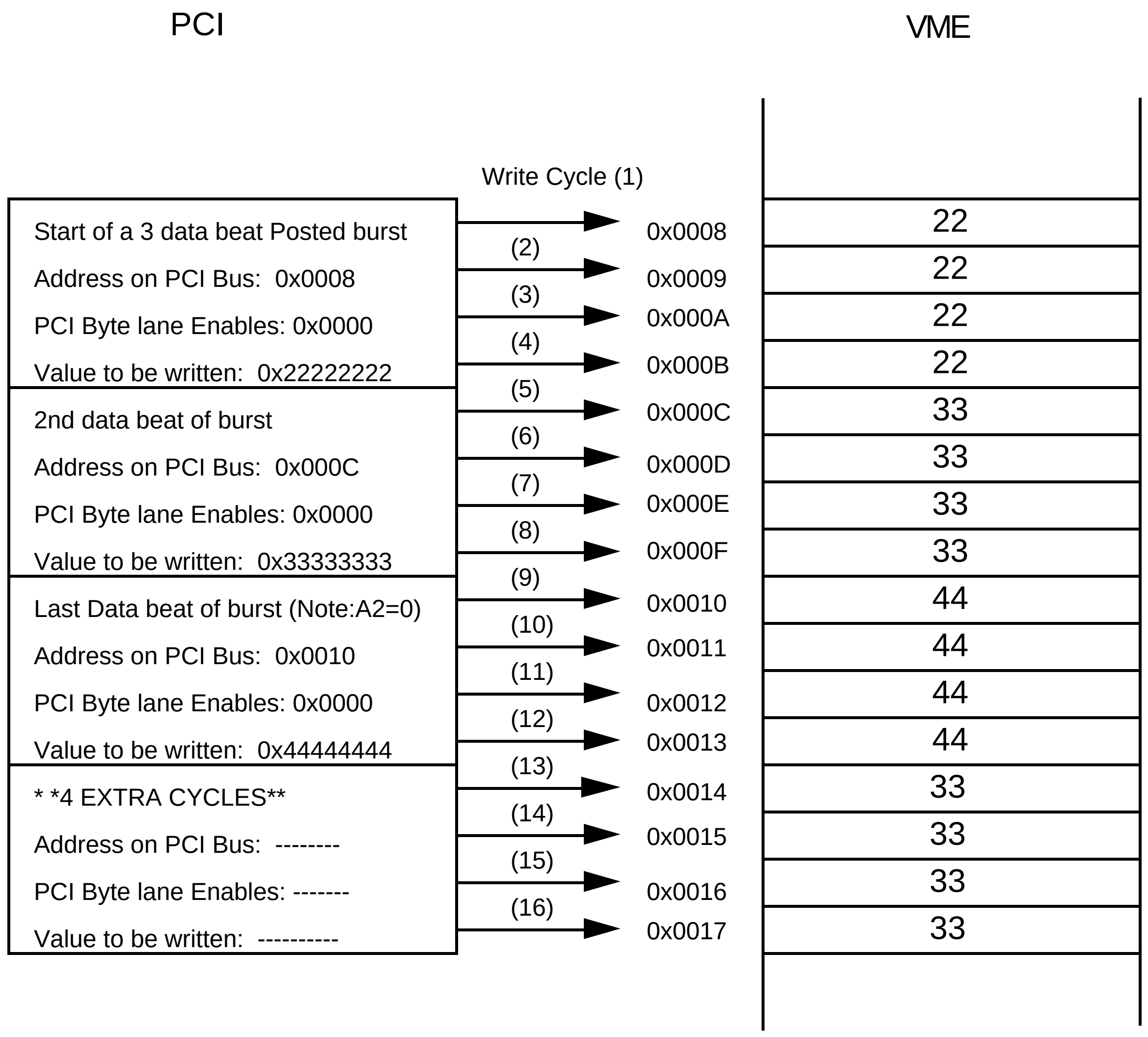
PCI
VME
Write Cycle (1)
Start of a 3 data beat Posted burst
Address on PCI Bus: 0x0008
PCI Byte lane Enables: 0x0000
Value to be written: 0x22222222
2nd data beat of burst
Address on PCI Bus: 0x000C
PCI Byte lane Enables: 0x0000
Value to be written: 0x33333333
Last Data beat of burst (Note:A2=0)
Address on PCI Bus: 0x0010
PCI Byte lane Enables: 0x0000
Value to be written: 0x44444444
* *4 EXTRA CYCLES**
Address on PCI Bus: --------
PCI Byte lane Enables: -------
Value to be written: ----------
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)
(14)
(15)
(16)
0x0008 22
0x0009 22
0x000A 22
0x000B 22
0x000C 33
0x000D 33
0x000E 33
0x000F 33
0x0010 44
0x0011 44
0x0012 44
0x0013 44
0x0014 33
0x0015 33
0x0016 33
0x0017 33

Figure 1.4: All Byte lanes enabled ---D32
(Zero translation offset; VME Maximum data width is 8 bits)

9. DMA WRITE TRANSACTION

Description

The Universe provides an internal DMA controller for data transfer between the PCI and VME bus. If all the below conditions are met, the Universe's DMA channel will stop processing data, but will not relinquish ownership of the VME bus. With VMEbus ownership locked at the DMA, neither the interrupt channel, nor PCI transactions will be processed to the VMEbus. In normal encounters of a VME bus error, the Universe will generate an interrupt. However, no interrupt will be generated for this situation.

The conditions that must be met for this problem to occur are:

- the Universe is processing a DMA write transaction to the VMEbus
- the DMA FIFO contains one or no entries, and
- a VME bus error occurs

Because the conditions rely upon the DMA FIFO being at a certain level, the probability of the errata occurring is a function of the fill vs. drain rate, in the DMA FIFO.

Impact

If these conditions surface, the Universe holds the VXBBSY line asserted halting all VME traffic in the system. The only way to release the lock-up on the VMEbus is by resetting the Universe. As well, since all activity on the VME bus is stopped, the CPU will be indefinitely retried on coupled accesses to the VMEbus, and on write accesses once the TxFIFO has filled up.

Solution

This problem occurs as the DMA FIFO empties. It is therefore possible to reduce the probability of this error by utilizing two fields found in the DGCS register. By decreasing VON, or increasing VOFF, the DMA FIFO will have a greater number of entries in its queue reducing the likelihood of there being fewer than two entries in the FIFO when a bus error occurs. The system, however, would suffer a performance loss.

The lock-up condition can be detected through application of a time-out on the VXBBSY signal. This signal can be monitored through external circuitry to detect whether the signal has been asserted too long. The length of time will depend on the settings of the VON field in the DGCS register, the PWON field in the MAST_CTL register and other system dependent parameters. When the time-out occurs, the CPU can be notified through an interrupt, or alternatively the Universe may be directly reset through the VRSYSRST# or RST# inputs.

A software time-out can also be implemented to monitor whether DMA operations are taking too long; again a system dependent parameter. Once the time-out has been detected, the CPU can perform a reset to clear the condition.

For applications in which the DMA is used to poll for the presence of VME slaves, DMA read operations may be used to avoid the problem. If DMA writes are required, set the DTBC field in the

DGCS register to 24 bytes. By programming the DMA to move at least 24 bytes, the user will guarantee that two entries remain in the DMA FIFO when the bus error occurs.

10. LOCAL SLAVE CHANNEL READ MODIFY WRITE CYCLES

Description

The Universe is designed to generate Read-Modify-Write (RMW) cycles on the VMEbus, through the use of its internal special cycle generator. During this cycle, however, incorrect VMEbus read data may be returned to the initiating PCI master. There are two manifestations of this problem.

When the RMW is performed to a 64-bit unaligned address (i.e. A2=1), the Universe will return inaccurate data to the CPU and the data it subsequently writes to the VMEbus will be wrong.

For 64-bit aligned RMWs (i.e. A2=0), the Universe may perform an inaccurate cycle on the VMEbus if the VME cycle performed by the Universe immediately prior to the RMW was not of the same data width and alignment. For example, an 8-bit RMW at address 0x01 will complete correctly only if the cycle immediately prior was also an 8-bit cycle (read or write) to the same address. If this condition is not met then both the data returned to the CPU and the data written to the VMEbus will be corrupted.

Impact

A Read-Modify-Write cycle to the VMEbus may result in incorrect data being written to the destination VME address. Additionally, the data returned to the CPU may be incorrect. The Universe still correctly accepts VME RMW operations as a VMEbus slave as documented in the Universe manual.

Solution

The effects of this errata may be avoided by making use of LOCK operations as recently defined in the VME64 specification and as documented in the Universe manual (ADOH cycle). However, many existing VMEbus slaves do not yet accept LOCK operations. For those that don't, two work-arounds are proposed. The first does not cover RMW operations where A2=1, while the second is more global in scope.

I.

The following is a suggested work around for the case in which the VME destination address contains A2=0. It does not cover the A2=1 case. A dummy PCI cycle to the destination VME address is used to set-up the Universe so that the subsequent VME RMW cycle operates correctly.

1) Initialize the following RMW registers as outlined in the Universe documentation. (SYC_ADDR, SYC_EN, SYC_SWP, SYC_CMP)

2) Complete a coupled PCI read/write from the SYC_ADDR using data width as that which will be used for the subsequent RMW. This dummy cycle internally sets up the Universe for the RMW cycle.

3) Complete the RMW operation as outlined in the Universe documentation by initializing the SYC_CTL register to enable the cycle and then performing a read to the destination address to initiate the RMW cycle.

The Universe's VME master interface is a shared resource between the DMA, the interrupt channel, and potentially other PCI masters. There is, therefore, a possibility that one of these resources could take over the VME master between the PCI dummy cycle and the intended RMW cycle resetting the internal set-up performed by the dummy cycle. To minimize this probability, the following suggestions could be implemented.

- Use the PCI LOCK# signal to lock the Universe against other PCI masters.

- Disable the DMA and Interrupt channels temporarily until the RMW transaction is completed to prevent these channels from gaining access to the VMEbus between the dummy cycle and the RMW, or

- Enable the coupled window timer, in the LMISC register. This will cause the Universe to hold ownership of the VMEbus, on behalf of the PCI slave channel, after processing a coupled transaction. (When the timer expires the PCI slave channel releases the VME master interface).

II.

The following suggestion involves using software to manually mimic the Universe's special cycle generator. When implemented, it will work for all VME destination addresses irrespective of their A2 value. This workaround may introduce a performance loss, and would fail to pass the RMW cycle across a remote bridge. In other words, it will only lock the resource as far as the slave VME interface.

The scenario is illustrated below.

1) Enable the VOWN bit, in the MAST_CTL register. This ensures that the Universe will have exclusive access to the VMEbus, preventing other VME masters to gain ownership.

2) Use the PCI LOCK# signal to lock the Universe against access by other PCI masters. This will not be necessary if there is only a single PCI master that will initiate accesses out to the VMEbus

3) Perform a read to the VME variable.

4) Process data and perform internal software manipulations.

5) Perform a coupled write to the VME variable.

6) Disable the LOCK# and the release VMEbus ownership by clearing the VOWN bit.

11. PCI BUS SLAVE IMAGE 0

Description

If PCI Slave Image 0 (LSIO) is enabled before a PCI reset (assertion of the Universe RST# pin) occurs, then LSIO remains enabled. Other LSIO parameters such as base address, bound address, and translation offset will change upon the reception of a PCI reset. A PCI reset can be caused by a VME reset (SYSRST*) or a s/w reset, if the h/w is designed to route the LRST# output to the RST# input.

The following fields in the LSIO registers do not revert to their reset states by a PCI reset:

EN, VAS, and LAS in the LSI0_CTL register,

Bits [31:28] in the LSI0_BS register, and

Bits [31:28] in the LSI0_BD register.

All other fields revert to their reset state. The reset states for these other fields are "0" except for the VDW field in the LSIO_CTL register.

Impact

If Local Slave Image 0 was enabled when a PCI reset is received by the Universe, the image will clear the low 27 bits of the base and bounds registers to zero, but the image will remain enabled. The implication is that the image may have moved to another location in PCI space potentially overlapping the image for another PCI device, another Universe slave image, or the Universe register image. The effect of overlapping another device's image will be erroneous and unpredictable PCI behavior. If it overlaps another Universe local slave image, an erroneous VME cycle will be generated matching the characteristics of the overlapped image. If it overlaps the Universe register image, accesses to local slave image 0 will result in register accesses.

Solution

(1) Program bits [31:28] in both the LSIO_BS and LSI0_BD registers to be the SAME VALUE. After a PCI reset occurs, the lower 28 bits of the base and bound addresses are cleared to zero but the top 4 bits [31:28] would remain the same. Since the base and bound addresses are the same, the slave image size effectively becomes zero, thus disabling this image. This workaround would limit the image size to a maximum of 256MB. Please refer to Fig. 1 below for an example.

Register Name	Before RST# Assertion	After RST# Assertion
LSI0_CTL	C0C2_510F	8082_0001
LSI0_BS	FFFF_FFFF	F000_0000
LSI0_BD	FFFF_FFFF	F000_0000
LSIO_TO	FFFF_FFFF	0000_0000

Table 1.1: Before and after effects of PCI Reset on PCI slave image 0

(2) Program the lower 28 bits in the LSIO_BS and LSIO_BD registers to zero and the LSIO_TO register to zero. After a PCI reset occurs, the base address, bound address and translation offset remain the same. However, the PWEN, VDW, PGM, SUPER and VCT fields in the LSIO_CTL register will revert to their default reset state. The image would be forced to have a minimum size of 256MB and a granularity of 256MB.

12. LOCAL SLAVE CHANNEL DISABLED BYTE LANE CYCLE

Description

The Universe's PCI slave channel is designed to handle posted burst transactions in which all byte lanes are disabled. However, under certain criteria, the Universe will generate an incorrect cycle on VME, and assert VME AS* forever. This will occur when all three of the following conditions are met:

- Posted (decoupled) burst write operation through the PCI slave channel
- Multiple data beat PCI transactions.
- A 64 bit aligned data beat (A2=A1=A0=0) is generated with zero enabled byte lanes, after the Universe has processed a data beat with enabled byte lanes within the same burst.

For these examples one data beat is defined as two 32 bit data phases.

Please refer to tables 1-3 for an illustration of how this problem is manifested.

(Note that all of the following examples use a 32 bit PCI data bus).

The case shown in Table 1 would result in a successful transfer. The second data beat is indeed 64 bit aligned but it contains enabled byte lanes.

PCI Address	Data Beat	PCI Data	PCI Byte Lanes (4 byte lanes)	Comments
0x0	1st	1	0000 (enabled)	Each data beat contains half of its byte lanes enabled. (Each byte lane represents one byte of data.)
		2	1111 (disabled)	
	2nd	3	1111	
		4	0000	

Table 1.2: Pattern A: Successful Transfer

The case shown in Table 2 would result in a successful transfer. The second data beat is 64 bit aligned and contains no enabled byte lanes. However, there was no previous data beat with byte lanes enabled.

PCI Address	Data Beat	PCI Data	PCI Byte Lanes (4 byte lanes)	Comments
0x0	1st	1	1111	Data beat with disabled byte lanes
		2	1111	
	2nd	3	1111	64 bit aligned, disabled byte lane data beat.
		4	1111	
	3rd	5	0000	Enabled byte lane data beat.
		6	0000	

Table 1.3: Pattern B: Successful Transfer

Table 3 demonstrates a failing transaction. The second data beat is 64 bit aligned and contains no enabled byte lanes. There was also a previous data beat with byte lanes enabled. In this case data value 5 would be placed at the wrong address and VME AS* would remain asserted.

PCI Address	Data Beat	PCI Data	PCI Byte Lanes (4 byte lanes)	Comments
0x0	1st	1	0000	1st data beat contains 4 enabled byte lanes.
		2	1111	
	2nd	3	1111	64 bit aligned disabled byte lane data beat
		4	1111	
	3rd	5	0000	Failure would occur
		6	0000	

Table 1.4: Pattern C: Failed Transfer

Impact

When all the conditions for this errata are met, **data will be transferred to an incorrect address** on the VMEbus. The Universe will also hold the VME AS* line asserted, **indefinitely, locking VMEbus ownership at the Universe**.

Solution

If possible, ensure that posted write transactions, with zero byte lanes enabled, do not follow the same byte lane patterns described in this errata. If these patterns cannot be avoided, disable decoupling through the Local Slave Channel.

13. VME SLAVE AND PCI TARGET CHANNEL DEADLOCK

Superseded by errata #19.

14. DMA WRITE CYCLES WITH PCI ALIGNED BURST SIZE=64 BYTES

Description

With PABS = 64 bytes, as defined in the MAST_CTL register (Table A.56), DMA transfers from PCI to VME will fail when the local start address ends with one of the following values: xx30, xx34, xx70, xx74, xxB0, xxB4, xxF0, xxF4. The failure could also occur during the process of a DMA transfer, in which the local start address was not one of the above addresses. This would occur if a PCI target disconnects in the middle of a 64-byte aligned burst and the DMA resumes its transactions on one of the above addresses.

The failure does **not** occur with DMA reads or with PABS = 32 bytes.

Impact

When the conditions specified in this errata manifest themselves, the Universe will read only up to 16 bytes from the PCI bus and place the contents in the TXFIFO. However the VME bus will never be requested by the Universe. It will remain locked in this state with no errors being logged and the DMA controller remaining active.

Solution

Program PABS = 32 bytes. The problem does not occur in this case.

15. VME INTERRUPT HANDLING

Description

As a VME interrupt handler, the Universe is designed to monitor any or all of the VMEbus interrupt levels simultaneously. However, in one particular scenario of events, the Universe could generate an IACK cycle on the VMEbus at an incorrect level. In addition, the interrupt channel could lock up preventing further generation of IACK cycles until the Universe is reset. This errata will manifest itself when all of the following conditions occur in sequence:

- The Universe receives an interrupt on one of its VRIRQ lines.
- The Universe, as an interrupt handler, requests the VME bus in order to execute its IACK cycle. (VXBR)
- The Universe receives another interrupt, at a higher level than the first, immediately before it executes its IACK cycle for the lower level interrupt. (VIACKOUT#)

Impact

When all the conditions for this errata are met, the Universe will respond in **one** of the following two ways:

1) The Universe will generate an IACK cycle on level zero, irrespective of the intended level. In addition, the Universe will not be able to handle further incoming VME interrupts until the next reset. The system could generate a VME bus error, lose a high level interrupt, and the ability to map VME interrupts to the PCI bus. All other activity including interrupt generation by the Universe remains operational.

2) The Universe will perform the higher level IACK cycle, but will store the status/ID in the register for the lower level interrupt. This would cause significant confusion for the PCI resource delegated to handle the interrupt. In this case the Universe will continue to handle further interrupts.

Solution

There are three possible solutions to this errata.

1) With the Universe's requester placed in Release When Done mode (RWD), a transparent latch can be used to gate the incoming IRQ lines. This gating technique will prevent the Universe from seeing an interrupt during the vulnerable error window. This hardware workaround will not work when the requester is in Release On Request mode (ROR). Please see Figure 1.5.

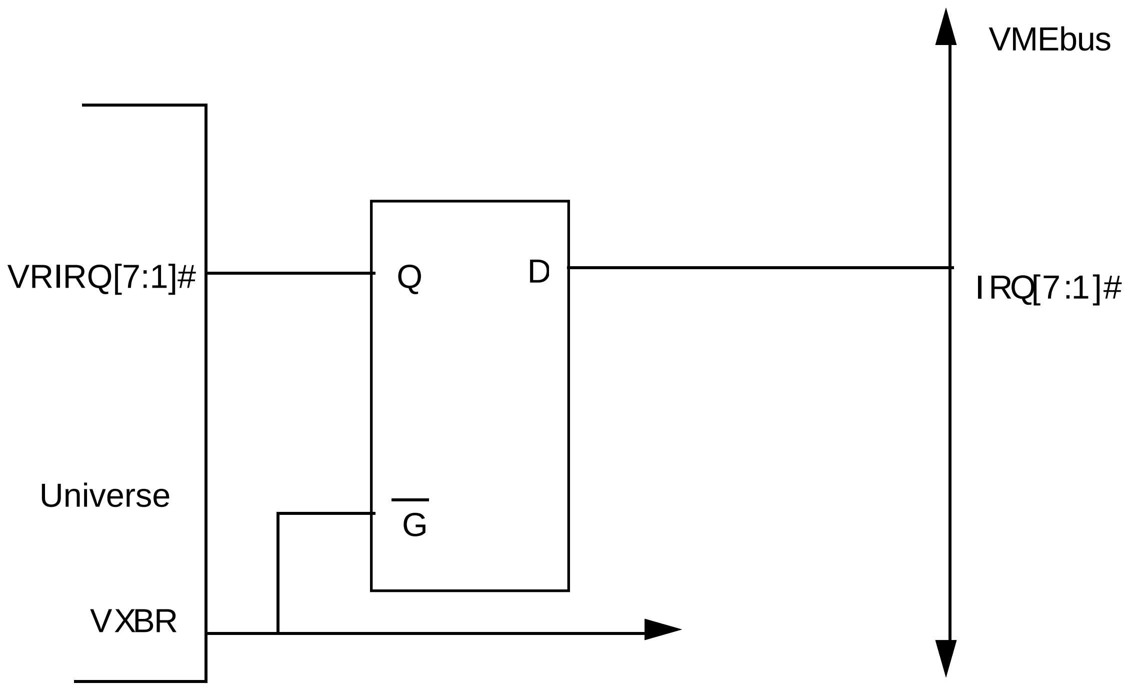

Figure 1.5: Release When Done IRQ workaround

2) Configure the Universe registers such that it will only accept VMEbus interrupts at one level.

3) The following suggestion, detailed in Figure 1.6, involves using a transparent latch, along with some logic gates, to prevent the Universe from seeing an interrupt during the vulnerable error window. Once implemented, it will work for both of the Universe's requester modes (Release on Request as well as Release on Demand). The latch should be transparent when the following is true:

i) IRQ[n]* is negated,
ii) (VXBBSY is asserted) and (VAS# is asserted),
iii) (VXBBSY is negated) and (VXBR is negated),
iv) (IRQ[n]* is asserted) and (IRQ[1 to (n-1)]* are negated.

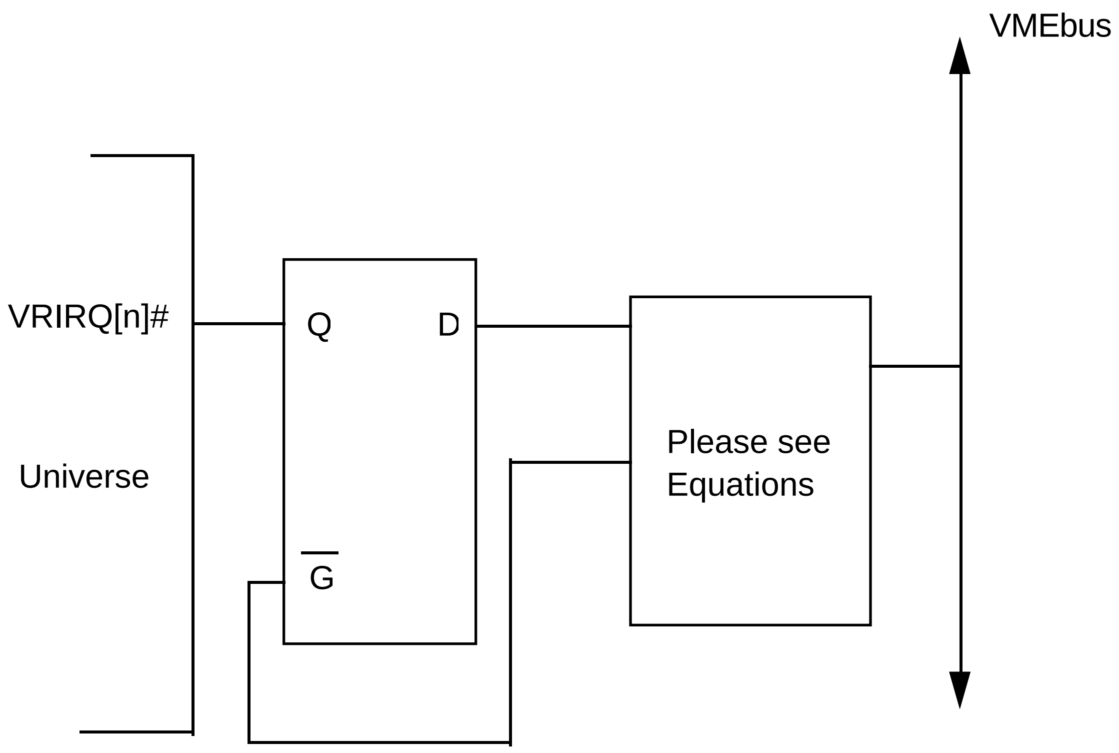

Figure 1.6: IRQ workaround for both requester modes

16. BGNT GLITCHES

Description

The Universe is designed to actively participate in VME bus arbitration handshaking. When the Universe receives a bus grant it will either accept bus ownership or pass it along to the next module. If the Universe receives an incoming bus grant and simultaneously makes a VMEbus request on the incoming level, it may pass along the grant for a few nanoseconds and then retract it. This results in a glitch being seen on the appropriate Universe VBGO#[3:0] line.

Impact

Depending upon the boards downstream from the Universe, the bus grant glitch could be interpreted as a bus grant, resulting in two simultaneous VMEbus masters.

Solution

This issue can be avoided by placing a capacitor (270pF suggested) on each of the VBGO#[3:0] lines being used for arbitration. The glitch has been observed to be between 2.5 and 5 ns in width.

17. COUPLED WINDOW TIMER AND RELEASE-ON-REQUEST

Description

After a PCI master completes a coupled cycle out to VME, the Universe will use its Coupled Window Timer (CWT) to determine how long to hold ownership of the VMEbus on behalf of the PCI slave channel. In the case of back-to-back coupled transactions, the CWT prevents the Universe from having to re-arbitrate for the VMEbus. However, in the case described below, the Universe may lock up the PCI resource, halting all PCI bus traffic. This errata will manifest itself when all of the following conditions occur in sequence:

i) A PCI master completes a coupled cycle through the PCI slave channel. The Universe resets its Coupled Window and holds on the VMEbus on behalf of the PCI slave channel,

ii) The CWT expires, and

iii) An external VME master requests the VMEbus while, at approximately the same time, the PCI master initiates a second coupled transaction.

When the Universe is in ROR mode, it gives up VMEbus ownership to the external VME master. However, the Universe may also fail to retry the PCI master initiated cycle, behaving as if it still had ownership of the VMEbus. In this case, the coupled cycle on the PCI side will never receive an acknowledgment from the Universe, and the PCI bus will be locked.

Impact

When the above conditions are encountered, activity on the PCI bus will halt until the coupled cycle, intended for the VMEbus is timed out by its initiating device. This issue will not manifest itself if the Universe is placed in Release When Done mode.

Solution

This issue can be avoided by using the Universe in Release When Done mode.

18. LSC’S COUPLED REQUEST TIMER

Description

When a PCI master initiates a coupled cycle out to VME, it is retried until the Universe has assumed VMEbus ownership. The Coupled Request Timer (CRT) is used to time out the Universe's ownership of the VME bus should the PCI master not retry its access.

The VME specification requires that devices continue to request VMEbus ownership until a bus grant has been issued. Once the grant has been issued, the device drives BBSY and releases its request. If the CRT times out, the Universe correctly holds its request active. However, upon eventual receipt of the bus grant, the Universe will not drive BBSY. Should a subsequent access to VME not occur quickly, the VME arbiter will time out, and the Universe will release its bus request.

Impact

In high traffic systems where the Universe may not be granted the VMEbus quickly, or where a PCI master may not have an opportunity to retry a cycle quickly, the VMEbus arbiter may encounter a arbitration time-out.

Solution

This issue can be avoided by either disabling the CRT, or by increasing it's value to a point where either the VMEbus will be granted before the timer expires, or within a margin where a PCI cycle is likely to retry its access.

19. VME MASTER OPERATION AFTER VME SLAVE FIFO FULL

Description

The Universe is designed to execute posted write cycles through its VME slave channel. During these cycles, the Universe queues data into its internal FIFO (RXFIFO). When the FIFO is filled, the Universe will hold DTACK* asserted. DTACK will remain asserted until an entry is dequeued onto the PCI bus.

If the Universe requests and becomes the VME master immediately after that FIFO full condition, and before any entries have been dequeued onto the PCI bus, the erroneous behaviour, described below, may occur.

<u>IACK Cycles</u>

If the Universe becomes the VME master and prepares to generate an IACK cycle while it is still asserting DTACK* (due to the FIFO full condition), the address of the last entry posted into the RXFIFO becomes corrupted. The address is substituted with 0x320. The IACK cycle continues on the VME side as normal once DTACK is negated by the Universe.

<u>Coupled Local Slave Channel Cycles</u>

If an external PCI master attempts to perform a coupled cycle through the Universe and the Universe gains ownership of VME while the Universe's is still asserting DTACK* (due to the FIFO full condition) a deadlock could occur . The VME side of the cycle cannot begin because the Universe has not released the DTACK* signal. DTACK* will only be released once there is room in the RXFIFO, however, this room cannot be created until the PCI initiated cycle completes, freeing up the PCI bus. The result is a deadlock condition hanging both VME and PCI buses.

Impact

When all the conditions for this errata are met, one of the following will occur:

<u>IACK Cycles</u>

During a posted write cycle through the VME slave channel, the Universe will write one or two data beats (depending upon the width of the VME cycle that was posted to the FIFO), to incorrect addresses. The VME IACK cycle, which overlapped the Universe's asserted VDTACK# signal, will execute normally.

Coupled Local Slave Channel Cycles

The Universe's VME slave channel will assert VDTACK# indefinitely. Ownership of VME is held by the Universe, while ownership of PCI is held by the external PCI master. Both VME and PCI buses will be locked up indefinitely, halting all traffic on the system.

Solution

Both manifestations of this errata can be solved by implementing the hardware schematic shown in Figure 1.8.

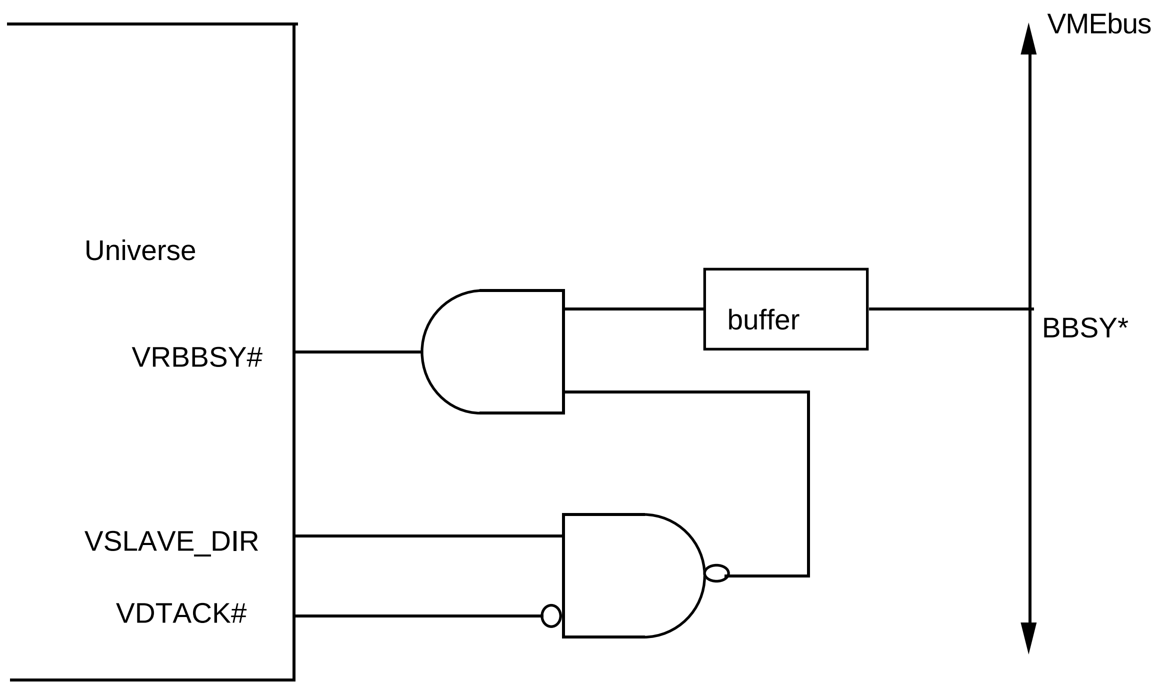

Figure 1.7: Errata #19 workaround

The following is a software solution to prevent the deadlock condition in which the Universe will lock up both the PCI and VMEbus.

Monitor the VME slave channel's FIFO before starting a coupled cycle through the PCI target channel in the following sequence: (Software solution)

i) Enable the VOWN interrupt through the LINT_EN and LINT_MAP0 register,

ii) Set the VOWN bit in the MAST_CTL register,

iii) Wait for VOWN interrupt to occur on PCI,

iv) Poll the FIFO empty bit from the PCI side (RXFE in MISC_STAT register) to ensure that VDTACK# has been released,

v) Perform the PCI initiated coupled cycle,

vi) Release the VOWN bit and clear the VOWN interrupt.

II. UNIVERSE MANUAL ERRATA

1. FIGURES 4.1 AND 4.2 LABELED INCORRECTLY

The labeling of Figures 4.1 and 4.2 in the Universe manual is ambiguous. These figures were placed in the "Signals and DC Characteristics" section of the manual with the understanding that they were to be used to map pin numbers to signal names, and not as a reference for layout.

The use of these figures for layout purposes will result in incorrect board design.

2. INCORRECT SYSFAIL AND ACFAIL SIGNAL TYPE SPECIFICATIONS

Both the SYSFAIL and the ACFAIL bits are labeled incorrectly in Table A.41 on page A-49. These bits are not read-only, but (like the others in Table A.41) R/Write 1 to Clear.

3. USE OF REQ64#

Old versions of the Universe manual (green Newbridge Microsystems editions) incorrectly stated that the Universe has an internal pull-up for the REQ64# pin.

In the most current manual, the description of the proper use of the REQ64# line in sections C.2 and 2.3.1.1 is contradictory. In order to have the Universe configured as a 64-bit PCI device, the REQ64# line must be pulled low at power-up and reset. For a 32-bit PCI configuration, this line must be pulled high.