MROMB Design Considerations Using the Intel® 80303 I/O Processor

Application Note

September 2001

Order Number: 273593-001

Contents

1.0 Introduction

The purpose of this document is to provide to system design engineers the design considerations to take into account when designing Modular RAID on Motherboard (MROMB)[1] solutions using the **Intel® 80303 I/O Processor (80303 IOP)**. This application note will address both system hardware and system software modifications and considerations necessary to implement a MROMB solution utilizing the 80303 IOP.

The following is an additional document that is referenced in this application note.

Intel® 80303 I/O Processor Developer's Manual	Order No. 273353-001

1.1 Document Organization

This document is divided into three main sections, each with its intended audience.

Section 2.0, "80303 I/O Processor Architecture Overview", gives the reader an overview of the dual PCI bus architecture of the 80303 IOP.

Section 3.0, "Hardware (Motherboard) Design Considerations", provides the system level H/W designer with the interrupt routing details and external logic required to control the IDSEL signal to implement the MROMB application using the 80303 IOP.

Section 4.0, "Software (System Resources) Design Considerations", provides the system level S/W designer with the software considerations required to implement MROMB solutions.

In addition to Section 5.0, "Managing the Embedded I/O Controller", a review of the implementation, there are appendices that address particular issues related to MROMB. Appendix A, "Microsoft WHQL Considerations", addresses potential WHQL certification considerations. Appendix B, "RAIDIOS Schematic", gives a schematic diagram of RAIDIOS circuitry.

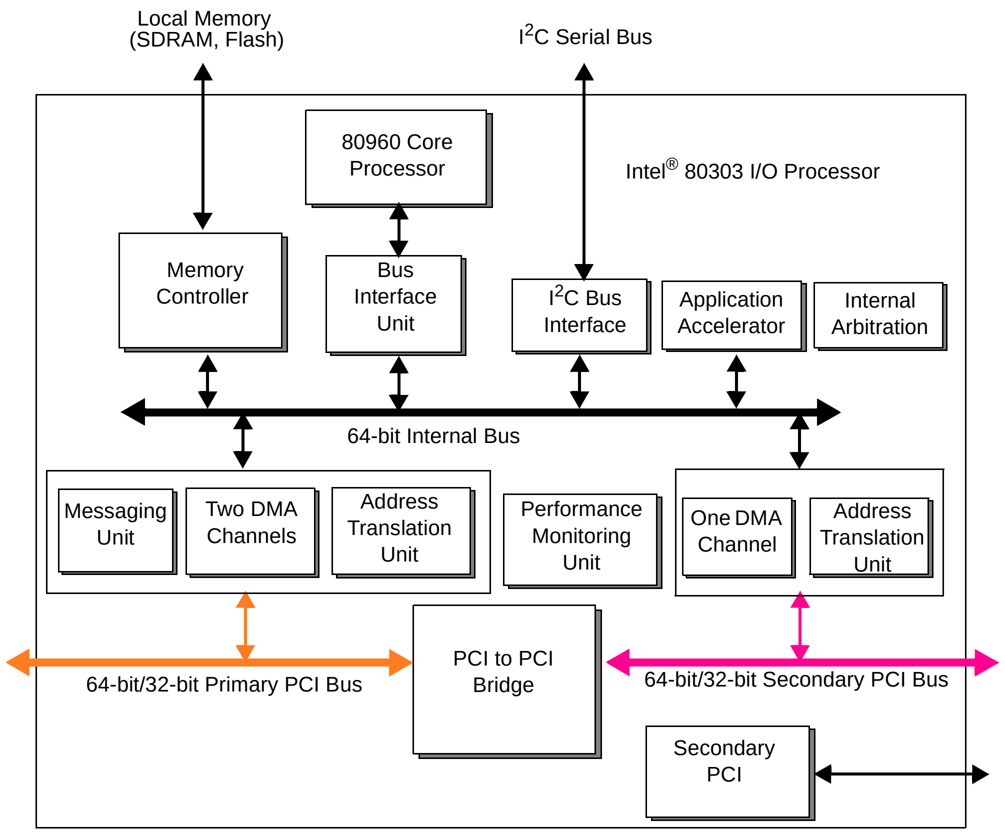
1. MROMB is also often referred to as Zero-Channel RAID (ZCR)

2.0 80303 I/O Processor Architecture Overview

Before getting into the MROMB design considerations, let's briefly review the dual PCI bus architecture of the 80303 IOP.

The 80303 I/O processor implements a dual PCI bus architecture through an on-board PCI-PCI bridge. For each PCI bus, Address Translation Units (ATU) and Direct Memory Access (DMA) units provide communication and data movement capabilities between the applicable PCI bus and the I/O processor core and local memory (flash or SDRAM). The dual PCI bus architecture diagram of the 80303 I/O processor is shown in Figure 1.

Figure 1. 80303 I/O Processor Functional Block Diagram



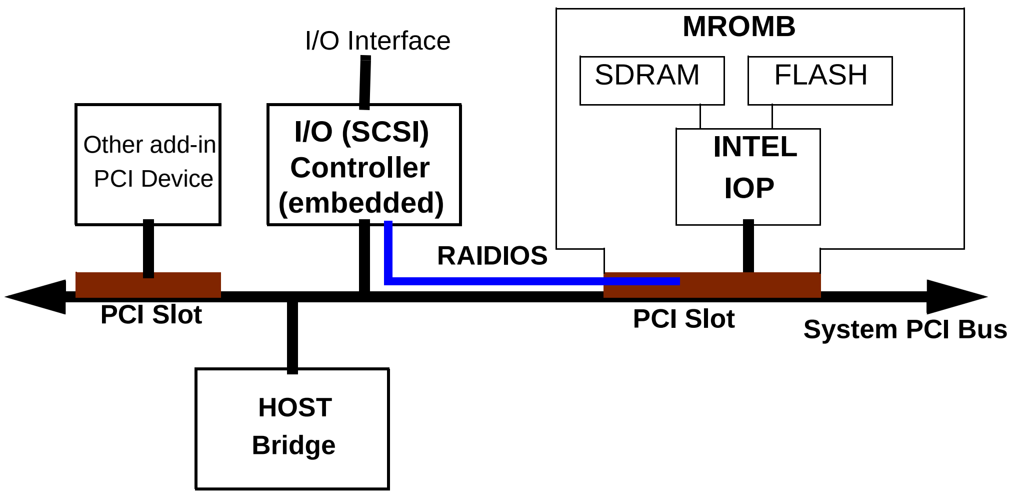
Detailed information on this I/O processor can be found in the *Intel® 80303 I/O Processor Developer's Manual.*

2.1 80303 IOP Architecture and MROMB

The architecture of an MROMB implementation affects system resource allocation in that the PCI-PCI bridge and secondary PCI bus of the 80303 IOP are not used. The 80303 IOP communicates with the I/O controller via the 80303 IOP's primary PCI bus which is visible to the host system. For the IOP to be able to first configure the I/O controller as a component of the MROMB solution, and then service the interrupts of the I/O controller, a mechanism must be put into place to isolate the I/O controller from the host system. This mechanism is the ***RAIDIOS (RAID I/O Steering)*** circuitry, which is required to be integrated onto the system motherboard.

In addition to RAID firmware considerations, the remaining sections of this document concentrate on the RAIDIOS circuitry and the logic required to be designed onto the system motherboard to enable a standard PCI slot to detect when an MROMB add-in adapter is inserted into the slot. This circuitry allows the embedded I/O (e.g. SCSI) controller to function either as a standalone I/O controller (the MROMB add-in adapter is not inserted in the RAIDIOS-enabled PCI slot) or as one component of the complete RAID solution when the MROMB add-in adapter is inserted into the RAIDIOS PCI slot.

Figure 2. MROMB System Architecture



3.0 Hardware (Motherboard) Design Considerations

For a motherboard/system to function with a MROMB add-in PCI adapter, one PCI slot must be specially wired with the RAIDIOS circuitry. RAIDIOS is comprised of two fundamental components, ***IRQ steering*** and ***IDSEL (Initialize Device Select)*** control. IRQ steering refers to re-routing the I/O controller’s IRQ line so that when the MROMB adapter is present, the IRQ signal is routed to the PCI slot. When the MROMB adapter is not present, it is routed as if it were a standalone I/O controller (to the *System Interrupt Controller*). *IDSEL control* allows the MROMB adapter to take control of the I/O controller. By inserting a switch on the I/O controller's IDSEL line, the MROMB adapter can effectively hide the controller from the host and use it as part of its own RAID subsystem.

On the system side, the system design engineer must take care to ensure that a specific PCI address range, to be used by the I/O controller as configured by the 80303 IOP, does not cause system issues, as it is not host configured.

3.1 Interrupt Routing

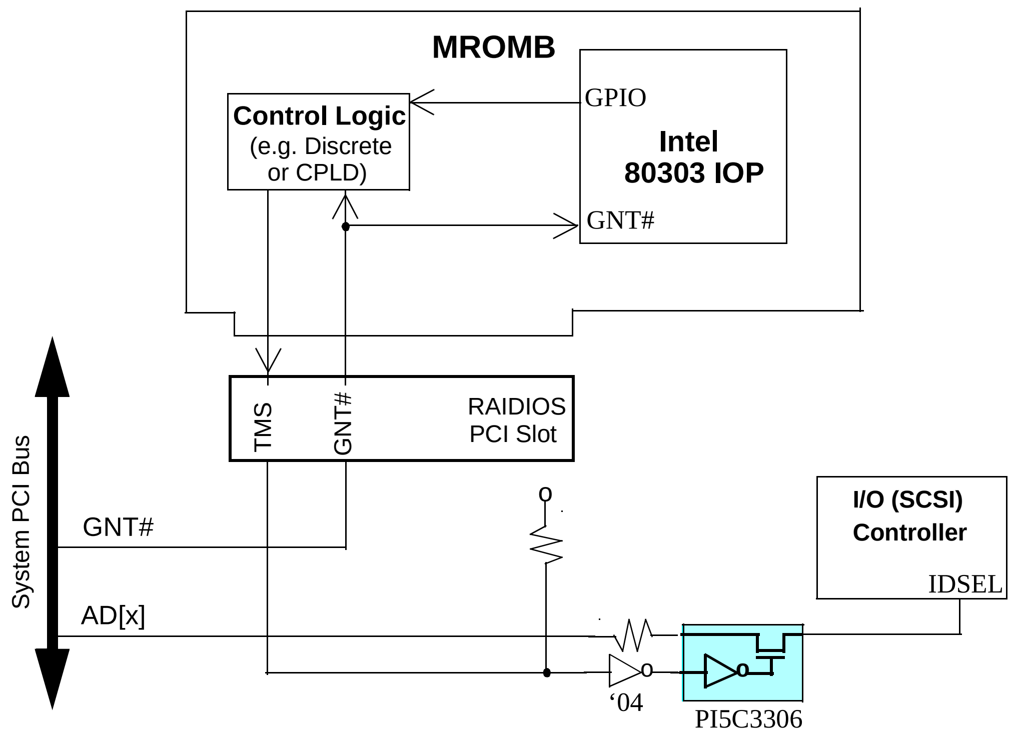
In the standard implementation of an onboard PCI I/O controller, its interrupts are routed to the *System Interrupt Controller.* Any interrupt from the I/O controller requiring service by the device's host driver will be received by the host interrupt controller and serviced appropriately. To enable MROMB, the I/O controller's interrupt signal(s) must be routed to the RAIDIOS-enabled PCI slot's INTC and INTD lines when the MROMB card is present in the appropriate PCI slot. This is accomplished using the RAIDIOS circuitry on the motherboard. Inserting the MROMB adapter into the RAIDIOS-enabled slot and powering on the system causes the JTAG signal TDI to be tied to low. When TDI is pulled low, the tri-state buffers, connected to TDI, are enabled. This reroutes the INTA and INTB signals from the I/O controller to INTD and INTC of the 80303 IOP located on the MROMB adapter. This allows the IOP to service the interrupt requests of the I/O controller. Removing the MROMB adapter disables the tri-state buffers, which returns the servicing of the I/O controller's interrupts to the *System Interrupt Controller.* Refer to Appendix B, “RAIDIOS Schematic” for a schematic view of this implementation.

3.2 Initialize Device Select (IDSEL) Control

When the system boots after a reset, the host BIOS initiates a PCI bus scan to find all of the PCI components installed in the system. The system uses the IDSEL signal to identify the I/O device and then assign the necessary resources. By taking control of IDSEL, the MROMB adapter's IOP executes configuration cycles to the I/O controller and properly hides it from host initiated configuration cycles. Care must be taken to ensure that IDSEL for the I/O controller is never enabled unless the 80303 IOP has been granted the bus for the PCI configuration cycle. One solution could be to have the MROMB adapter use its own GNT# line with some ***control logic*** (a CPLD or discrete logic can be used) to ensure that IDSEL for the I/O controller is only enabled when the MROMB adapter has control of the PCI bus. This ensures that the host will not see the I/O controller in a PCI bus scan scenario. See Figure 3, “MROMB IDSEL Block Diagram”, for a graphical presentation of this implementation.

When the MROMB adapter is present in the RAIDIOS-enabled PCI slot, the JTAG signal, TMS, is driven low thereby hiding the I/O controller from the system via steering logic on the motherboard. In order for the 80303 IOP to configure the I/O controller as a component of the MROMB solution, the I/O controller must be unhidden (it’s IDSEL is enabled). To ensure that only the IOP can detect the I/O controller during PCI configurations it must be unhidden only when the 80303 IOP has been given control (GNT#) of the bus. When the 80303 IOP located on the MROMB adapter performs configuration cycles on the PCI bus segment that it shares with the I/O controller when it has control of the PCI bus (GNT#), TMS is driven high to enable IDSEL of the I/O controller. When the 80303 IOP is finished configuring the I/O controller it relinguishes control of the bus (no longer has GNT#) and TMS is again driven low, disabling IDSEL of the I/O controller. This implementation allows only the 80303 IOP to control the I/O (SCSI) controller during PCI configuration cycles and to setup the necessary resources in host memory for the I/O controller. Refer to Appendix B, “RAIDIOS Schematic”, for a schematic view of this implementation.

Figure 3. MROMB IDSEL Block Diagram



3.3 PCI Interface Connector Requirements

For the MROMB implementation as described in this application note, a standard PCI connector compliant with the PCI Local Bus Specification Revision 2.2 is required for the PCI slot to also be compatible with standard PCI adapters other than the MROMB adapter. However, the MROMB solution architect may choose a proprietary slot designed solely for the MROMB adapter.

4.0 Software (System Resources) Design Considerations

4.1 Memory Allocation

With a standard RAID card, the I/O (SCSI) controller is located on the add-in card attached to the secondary PCI bus behind the PCI-PCI bridge of the 80303 IOP. This allows for the I/O controller to be configured in a private memory space. In a MROMB solution both the IOP and the I/O controller are located on the same PCI bus segment, and the MROMB adapter hides the embedded I/O controller. The architecture of an MROMB solution is not able to provide for the private memory address space that is required by the I/O controller for its internal resources. We discuss two possible solutions for this requirement; however, other solutions may exist.

The first option is to reserve space for the I/O controller within the BAR (Base Address Register) of the 80303 IOP. The IOP configures the I/O controller to respond to PCI transactions within this address range. Note that it is possible for the host to relocate the base address originally assigned to the IOP BAR. This would require the IOP to quiesce all activity and reprogram the I/O controller accordingly. To accomplish this an external interrupt must be triggered in the event that the IOP's BAR is written to. If a CPLD was used in the ***control logic***, this same CPLD could possibly be used to accomplish this. The RAID firmware must take the necessary steps to ensure that all of this works properly.

The second option takes advantage of the fact that the MROMB solution is already platform specific due to the RAIDIOS-enabled PCI slot requirement. The MROMB add-in adapter can be designed to configure the motherboard's embedded I/O controller so that the I/O controller's Base Address Register (BAR) is always mapped to a predefined address range in host address space. One possibility for this address range might be the top 1 MB of the 4 GB host address space provided that it can be ensured that using this space will not cause any negative behavior. Some system chipsets will decode this address range by default to the system BIOS flash hanging off an ISA bus. If the MROMB PCI slot is located on a different PCI segment and provided the chipset will not forward transactions in this range upstream before the hidden I/O controller can claim them, then there should be no system issues. Verification, to the designer's satisfaction, must also be made to ensure that the host will not reassign the chosen address range at a later time. As system memory maps are generally respected by host OS's this should be the case and not be an issue. It is important to note that in this solution the SCSI BAR is always mapped to this location by the IOP. The host system does not configure this BAR. It is therefore a requirement of the system design engineer to evaluate the target platform's architecture (chipset, BIOS, etc.) to ensure that when the I/O (SCSI) controller decodes addresses in the range chosen, this does not cause issues under any and all system operating environments.

5.0 Managing the Embedded I/O Controller

This section reviews the necessary steps involved with managing the embedded I/O controller in an MROMB implementation.

5.1 Discovering the I/O (SCSI) Controller

Two methods may be implemented for discovering the I/O controller(s) that is to be controlled by the 80303 IOP MROMB solution. The first method is to hard code the PCI slot values in the RAID firmware based on the hardware design of the MROMB implementation. The second method is by dynamic discovery during initialization.

Hard coded slot value(s) is the simplest to implement but lacks the flexibility to migrate the firmware to different platforms.

The dynamic discovery process provides the MROMB implementation with the greater flexibility. In this method, the RAID firmware can identify the I/O controller(s) that it needs to control by completing two PCI bus scans when the IOP has been given control of the PCI bus. First it completes a bus scan with GPIO low (i.e., the I/O controller's IDSEL is enabled), and then compares this first scan with a follow-up second bus scan conducted with GPIO high (i.e., the I/O controller's IDSEL is disabled). Note, that the bus scans can be done in either order with respect to the GPIO signal. GPIO controls IDSEL by enabling and disabling the TMS signal's ***control logic***. See Section 3.2.

The 80303 IOP has eight integrated general purpose input output (GPIO) pins, referred to as GPIO[7:0]. These pins, used in conjunction with the control logic, can be used by the 80303 IOP core to control the IDSEL input to the I/O controller. The output function of the GPIO pins is controlled by two registers as shown in Table 1 and Table 2.

The output enables are mapped on a per bit basis to each of the data bits in the GPIO Output Data Register. When a bit of the GPIO Output Enable Register is cleared, the corresponding data bit value in the GPIO Output Data Register will be actively driven on the appropriate GPIO pin. To enable a particular GPIO pin as an output following the de-assertion of P_RST#, a weak pull-down needs to be connected to the GPIO pin to overdrive the internal weak pull-up device.

Once the I/O controller(s) is identified, the firmware must then properly resource the targeted embedded I/O controller. Upon completion of all the necessary bus scans, GPIO is again toggled low, hiding the I/O controller from the system. The firmware then stops retrying the host-initiated configuration cycles and the system continues it's normal boot process. The 80303 IOP must make certain not to interfere with system configuration cycles. This can be done during board layout as described in Section 3.0 of this paper.

Table 1. GPIO Output Enable Register (address=0000 171Ch)

Bit	Default	Description
7:0	1	When clear, bit 7:0 of the GPIO Output Data Register is enabled onto the GPIO[7:0] pin.

Table 2. GPIO Output Data Register (address=0000 1724h)

Bit	Default	Description
7:0	0	This bit's value is driven on the GPIO[7:0] pin if bit 7:0 of the GPOE register is cleared.

5.2 Configuring the I/O Controller

RAIDIOS logic provides the mechanism to hide and unhide the SCSI device. Because the SCSI device is hidden from the host during a PCI scan, the MROMB firmware is responsible for the configuration of the SCSI device in the same manner in which the host would configure the device in the absence of the MROMB card. The only difference is that MROMB will always configure the SCSI device per one of the options described in Section 4.1, or an equivalent vendor unique method.

5.3 Conclusion

MROMB is a revolutionary new method of integrating intelligent RAID capabilities into a broader range of products. This application note explains the simple and economical way of preparing a system for MROMB (using the 80303 IOP) compatibility via the addition of some simple logic on the motherboard along with the verification of an address range usage. As with any design, Intel recommends that customers perform thorough validation of their application hardware and software prior to production.

Appendix A Microsoft WHQL Considerations

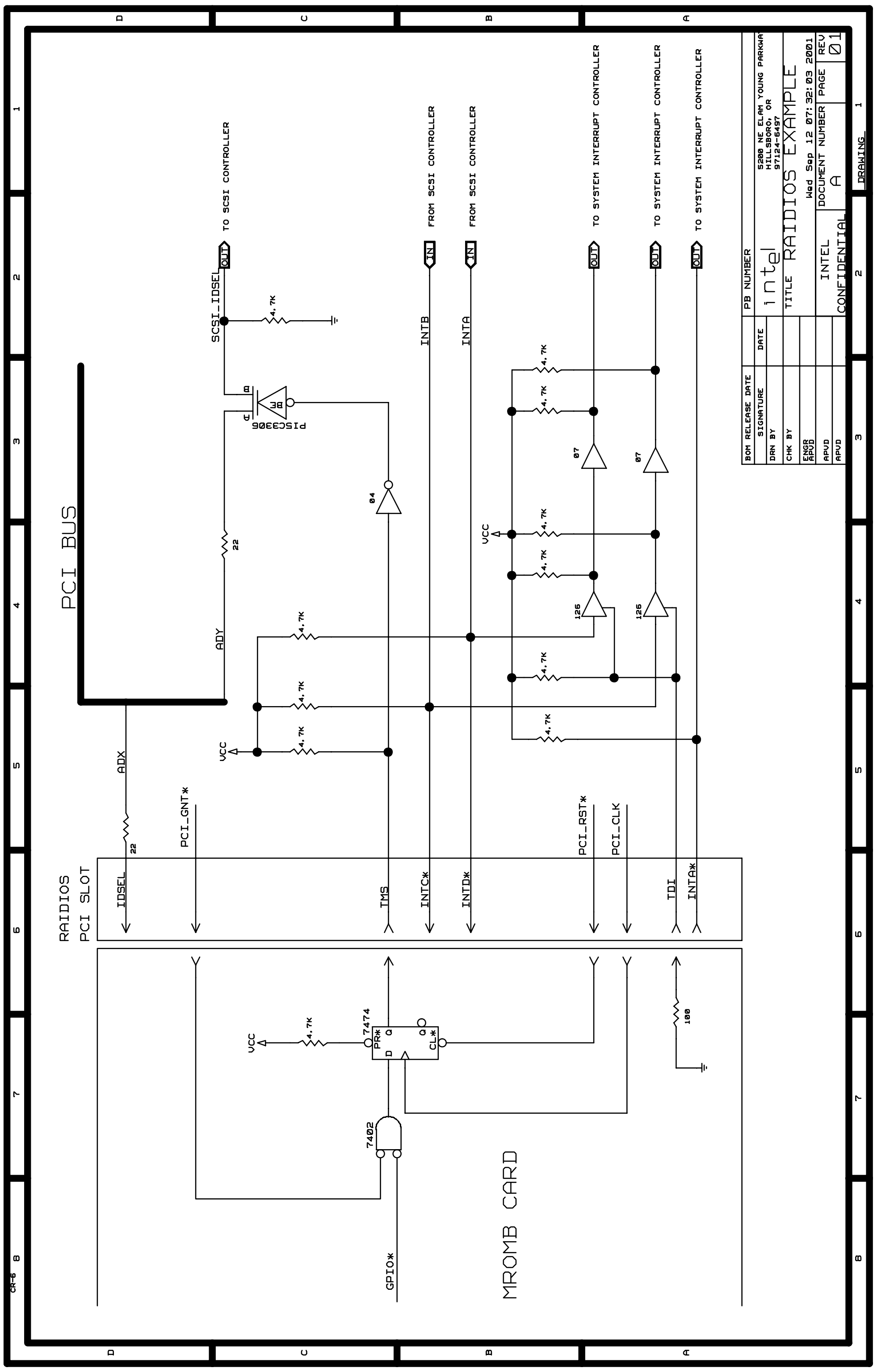
In the past, less complete architectures (e.g. SISL) identified various challenges in implementing zero-channel RAID and passing WHQL (Windows* Hardware Quality Labs) certification tests. This application note describes the necessary components that should prevent this MROMB implementation from being a roadblock to WHQL certification. The methods described in this application note have been prototyped, including the dynamic discovery of the IO device, and have passed the following Microsoft pre-certification tests for WHQL.

Note: Passing the pre-certification tests does not ensure that the pre-tested system would pass official testing but it is the best indication short of submitting a production system for testing. Intel does not guarantee that a customer's final implementation based on this paper will pass WHQL.

Table 3. WHQL Testing (Windows 2000) Using HCT Version 9.5 with 9.502 Update

Pass	Test Name	Description
√	Memory AWE (Server Test Suite)	The AWE test tests the Win32 Address Windowing Extensions on machines with more than 4 gigabytes (GB) of memory (RAM). The Address Windowing Extensions provide user applications with 32-Bit virtual addressing to greater than 32-bit regions of physical memory.
√	Syscache (Server Test Suite)	This test reads, writes, and verifies data with and without the Windows NT buffer feature. Testing halts when the SysCache test detects data corruption.
√	NTFS File IO (Server Test Suite)	This test uses various file I/O system calls to test the NTFS.
√	Stress Disk Test (Server Test Suite)	This test reads, writes, and verifies a 1MB pattern for 30 minutes by creating multiple threads for the drive you select. This ensures that the drive correctly reads and writes data. A drive partition can use any file system format and can include CD-ROM drives. A thread that tests a CD-ROM drive reads from, but does not write to, that drive.
√	Stress System Test (Server Test Suite)	This test puts an abnormal load on the system by simultaneously stressing disk access, memory management, GDI operations, module management and system caching.
√	NTFS File I/O (RAID Test Suite)	This test uses various file I/O system calls to test the NTFS.
√	Syscache (RAID Test Suite)	This test reads, writes, and verifies data with and without the Windows NT buffer feature. Testing halts when the SysCache test detects data corruption.
√	Device Stress (RAID Test Suite)	SdStress tests storage API commands and stresses the hard disks and CD-ROMs in a system.

Appendix B RAIDIOS Schematic


PCI BUS
RAIDIOS PCI SLOT
MROMB CARD
TO SCSI CONTROLLER
FROM SCSI CONTROLLER
FROM SCSI CONTROLLER
TO SYSTEM INTERRUPT CONTROLLER
TO SYSTEM INTERRUPT CONTROLLER
TO SYSTEM INTERRUPT CONTROLLER
OUT
IN
SCSI_IDSEL
INTB
INTA
PI5C3306
BE
ADY
ADX
VCC
4.7K
22
100
04
07
126
IDSEL
PCI_GNT*
TMS
INTC*
INTD*
PCI_RST*
PCI_CLK
TDI
INTA*
7474
PR*
CL*
D
Q
7402
GPIO*
PB NUMBER
TITLE RAIDIOS EXAMPLE
5200 NE ELAM YOUNG PARKWAY HILLSBORO, OR 97124-6497
Wed Sep 12 07:32:03 2001
DOCUMENT NUMBER
A
PAGE
REV
01
INTEL CONFIDENTIAL
DRAWING
BOM RELEASE DATE
SIGNATURE
DATE
DRN BY
CHK BY
ENGR APVD
APVD
APVD