FAULT DIAGNOSIS USING PROCESSOR-CONTROLLED TEST – A BASIC EXAMPLE

PRE-REQUISITES

It is assumed that the reader is familiar with the basic operation of our Processor-Controlled Test (PCT) product, and has a reasonable understanding of processor board architecture.

INTRODUCTION

This application note should be read in conjunction with Application Note #3. The following block diagram (Figure 1) illustrates a board architecture that has been simplified for the purpose of explaining how to locate faulty components when a board fails a PCT CPU run control test.

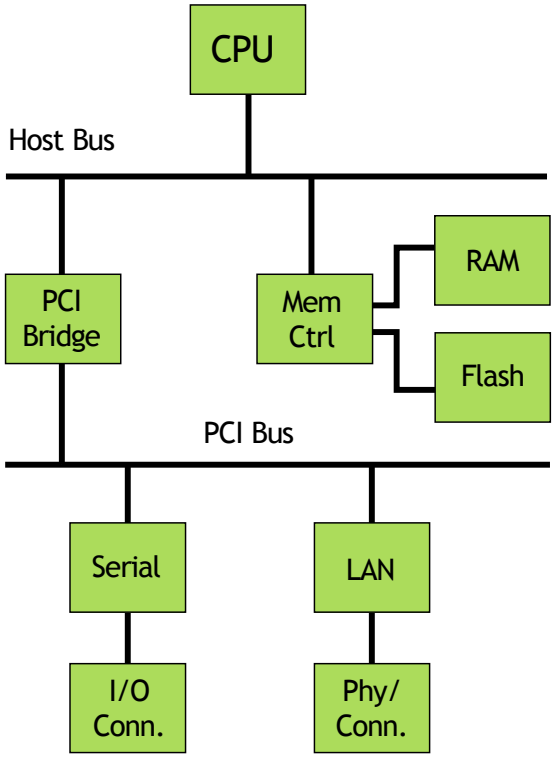

Figure 1 - Simplified Block Diagram.

With PCT tests begin at the CPU and work progressively outwards or downwards when considering the board's architecture as an inverted tree. A failure high up the tree, say for example in the PCI Bridge, will cause devices lower down the tree to fail PCT's tests. Repairing the failing PCI Bridge and re-testing the board will more than likely show that these lower devices are functioning correctly.

A good understanding of the board's architecture is therefore important in the diagnosis of faults. The block diagram in PCT's Operator and Developer modes provides useful guidance in fault location.

This application note explains how to decide in which functional block the fault lies:

- Processor Area (Section 2.1)
- Buses (Section 2.2)
- Bus Bridges (Section 2.3)
- Memory (Section 2.4)
- Input/Output (Section 2.5)

Having made that decision, the techniques described in Application Note #3 (Fault Diagnosis), for each of the above functional blocks, should be used to precisely locate the fault. **The above section numbers are the same in both documents.**

PARENT AND CHILD RELATIONSHIP

The terms “Parent” and “Child” are used in this application note to clarify the relationships of devices and buses in the block diagrams. In Figure 1, the CPU is the **parent** of the PCI Bridge and the Memory Controller. The PCI Bridge is a **child** of the CPU. The Host Bus is a parent bus of the PCI Bridge; the PCI BUS is a child bus of the PCI Bridge.

ASSESSING FAILURES IN THE BLOCK DIAGRAM

Figure 1 shows a simplified board with a CPU, a Memory Controller connected to RAM and Flash memory, a separate PCI Bridge ASIC, and a single PCI Bus with only Serial and LAN controllers on it. These controllers are attached to external I/O connectors (the LAN Phy is included in the same block as the connector). The equivalent block diagram displayed in PCT would have a slightly different layout, but the same blocks and buses would be present.

When a board test is run in PCT, if the test scripts that are associated with a specific block are successful, the block is shown in green. If any of the associated scripts fail, the block is shown in red. If another block has already failed, the current block is shown in pink (only the first failing block is red).

This system of green, red and pink diagnostic guidance relies on the fact that tests have been correctly developed:

- Test scripts associated with a specific functional block must test as much of the block as possible to locate any faults. If this is not done, the block may be shown in green, even though it could be faulty.
- The order in which the functional blocks are tested must start from the CPU and work systematically down the tree-like hierarchy. If the order is not correct, the first found failing device shown in red may be below other failing devices shown in pink. This would not help failure diagnosis.

In the following failure examples it is assumed that the tests have been correctly developed.

2.1. PROCESSOR AREA FAILURE

2.1.1. CPU Tests

Tests that would normally be associated with the CPU functional block include:

- Take control of CPU via its debug port, utilizing the JTAG interface (see section 2.1.3.1.).
- Check the JTAG ID Code of the CPU (see section 2.1.3.2.).
- Check register access (see section 2.1.3.3.).

2.1.2. Example Block Diagram Results

If the CPU failed, all devices below it would also fail, resulting in a block diagram as shown in Figure 2.

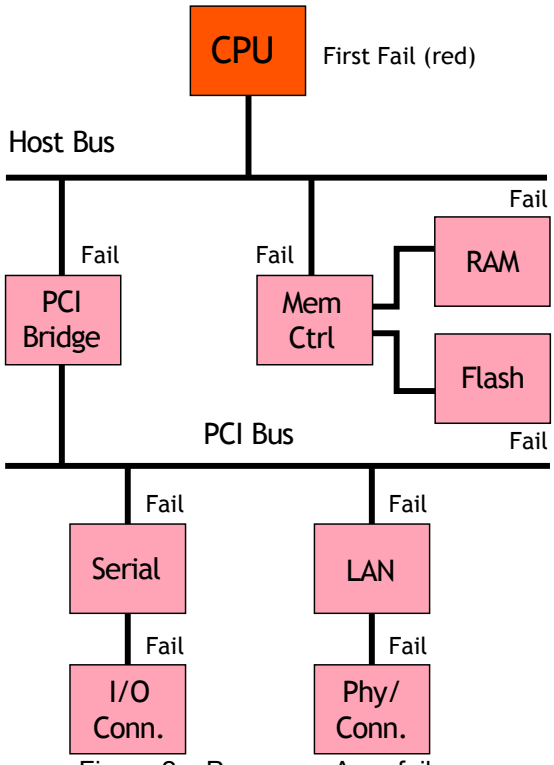

Figure 2 – Processor Area failure.

2.1.3. Possible Failures and Error Messages

2.1.3.1. Failure to Take Control of CPU.

If PCT fails to take control of the CPU via the JTAG interface the following messages could be displayed:

```
DEBUG PORT COMMUNICATIONS FAILURE: Failure configuring JTAG
```

In the case of a JTAG port dedicated to the CPU, the above message could signify:

a. Bad JTAG header connections or cabling to header
b. Bad JTAG lines to CPU (shorts, opens, etc.)

c. Bad CPU connection or socket
d. Faulty CPU

In the case of a common JTAG port with other devices on the same scan chain as the CPU, the message could signify points a. to c. above, and also:

e. Faulty devices on the scan chain interfering with the JTAG signals

```
DEBUG PORT COMMUNICATIONS FAILURE: Failure to enter debug mode.
```

The above message implies that the JTAG configuration has been successful because this occurs before entering debug mode. The cause could be due to:

a. Faulty lines in the CPU area, e.g. no CPU clock, or status lines
b. Bad CPU connection or socket
c. Faulty CPU

```
DEBUG PORT COMMUNICATIONS FAILURE: Failure configuring processor registers
```

```
DEBUG PORT COMMUNICATIONS FAILURE: Failure loading processor cache
```

The two messages above will only occur if a failure develops during the test. This is highly unlikely but it would probably indicate a bad CPU.

See Application Note #3 Section 2.1 for guidance on locating CPU-related faults.

2.1.3.2. Failure when checking the JTAG ID Code of the CPU.

The JTAG ID Code is verified by reading and comparing a CPU internal TAP register against known-good values using the "`IDCode`" command.

If the JTAG ID code does not match, possible error messages include:

```
ID CODE FAILURE:
Value Read 60080F56
Value Expected 50080F29
```

This would signify one of the following reasons for failure:

d. The internal ID Code register is faulty. The register access test 2.1.3.3. would confirm whether this was the cause of the failure or not.
e. The CPU is the wrong type. This reason cannot be relied on because different JTAG ID's have been allocated to new releases of the same processor.

```
ID CODE FAILURE:
Value Read FFFFFFFF
Value Expected 50080F29
```

This would signify that there is an issue with the JTAG interface. A good JTAG ID code is bound to have a few data bits not equal to 1. This error implies that the data on the JTAG interface will always be the same.

2.1.3.3. Failure checking register access.

This involves checking the connectivity from the debug port to the internal registers of the CPU, thus verifying the internal workings of the CPU.

Reasons for failure of this test include:

a. Faulty CPU.
b. Bad CPU connections to the board.
c. Faulty signals on lines connected to the CPU.

See Application Note #3 Section 2.1 for guidance on locating CPU-related faults.

2.2. BUS FAILURES

2.2.1. Bus Tests

A bus is tested by making write/reads across the bus from the parent device (CPU in Figure 3) to registers in child devices that are attached to the bus (PCI Bridge and Memory Controller in Figure 3).

Tests that would verify the Host Bus in Figure 3 are as follows:

- Check access to Memory Controller registers.
- Check access to PC Bridge ASIC controller registers.

2.2.2. Example Block Diagram Results

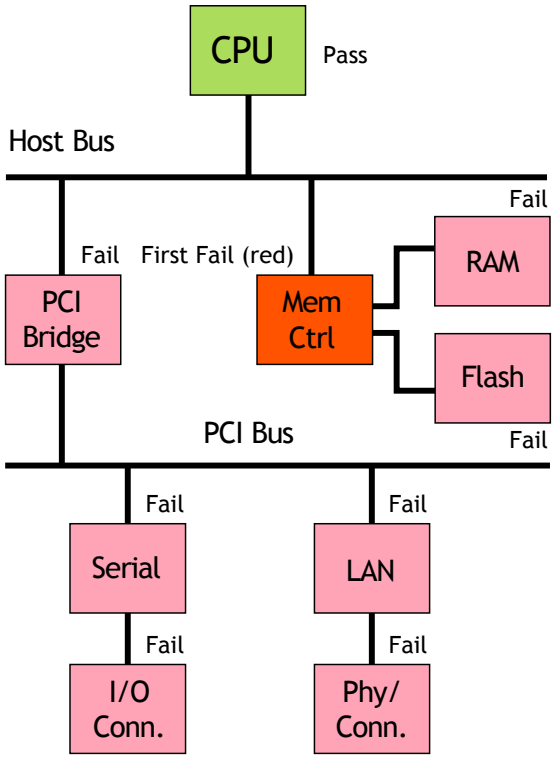

Figure 3 – Processor Area failure.

2.2.3. Possible Failures and Error Messages

ALL child devices attached to the Host Bus have failed (PCI Bridge and Memory Controller), **so the most likely cause is either a fault on the Host Bus or the back-end of the CPU**.

If a read of a controller register failed, the error message would be similar to:

```
FAILURE Controller Register:
Read     0765432112345678
Expected 8765432112345678
```

Converting the data value from hexadecimal to binary shows that data bit 63 is low when it is expected to be high. The following failures could be the cause:

2.2.3.1. Faulty CPU back-end.

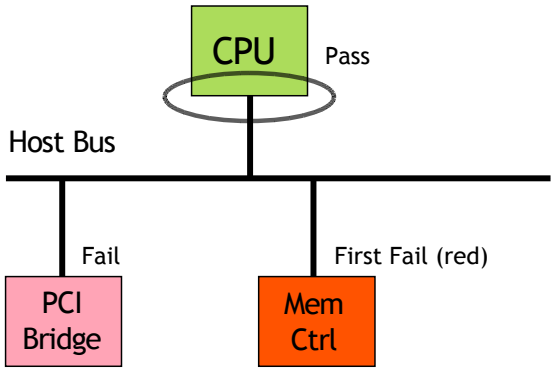

Figure 3a – CPU back-end failure.

Even though the CPU passed its register access tests and was displayed in green, the back-end of the CPU beyond the registers can only be tested by accessing the registers in a child device. So in this case, the CPU data bit 63 could be faulty; the information that should be communicated over the Host bus is not leaving the CPU (Figure 3a). The causes could be:

a. Faulty CPU.
b. Bad CPU connectivity. The CPU solder joints (or the CPU socket if fitted) are not connected correctly. This would lead to good data from the CPU not reaching the Host Bus.

See Application Note #3 Section 2.1 for guidance on locating CPU-related faults.

2.2.3.2. Faulty Host Bus.

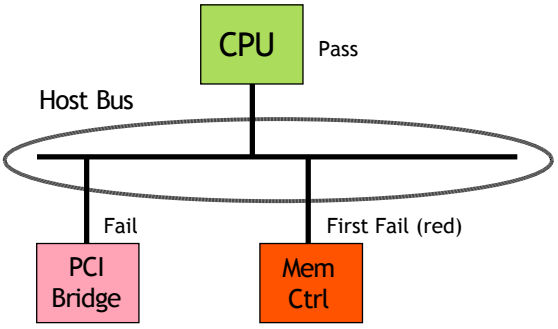

Figure 3b – Host Bus failure.

If the CPU back-end is not failing, the fault must be in data line 63 of the Host Bus (Figure 3b). The cause must be:

a. Host Bus interconnect fault. Data line 63 is stuck low.

If the error message for the memory controller had shown other faulty bits, possible causes would be:

a. Short circuit, open circuit, stuck high, or stuck low lines on the Host Bus.

See Application Note #3 Section 2.2 and Section 3.3 for guidance on locating Bus-related faults.

2.3. BRIDGE FAILURES

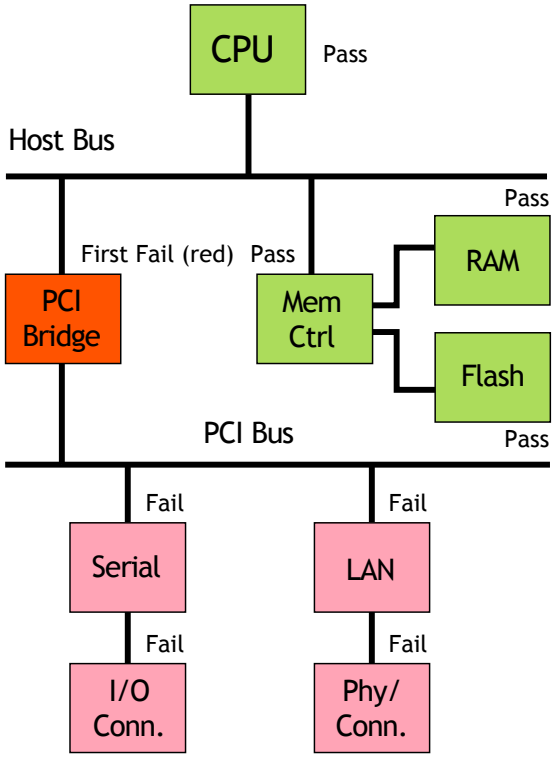

Figure 4 – Bridge failure.

2.3.1. Bridge Tests

The following tests are used to verify a Bridge:

- Check access to Bridge Controller registers.
- Set up registers for normal operation.

2.3.2. Example Block Diagram Results

Figure 4 shows that the PCI Bridge ASIC has failed, and all child devices attached to the PCI Bus have also failed their tests.

2.3.3. Possible Failures and Error Messages

This example is different from the previous example (2.2) because not all child devices on the Host bus have failed: the Memory Controller passed. This implies that the fault is not in the back-end of the CPU. **It is most likely to be a faulty PCI Bridge ASIC or an open circuit on the branch of the Host Bus leading to the Bridge. Stuck high, stuck low or shorted Host Bus lines are unlikely because these would probably have caused the Memory Controller to fail.**

Let's assume that when a register in the Bridge controller was read, the following error message was given:

```
FAILURE Controller Register:
Read     8765432112345678
Expected 0765432112345678
```

If we convert these data values from hexadecimal to binary, we find that data bit 63 is high when it was expected to be low. **It is unlikely to be a stuck high on the Host Bus because the Memory Controller passed**. The following failures could be the cause:

2.3.3.1. Host Bus Interconnect Fault.

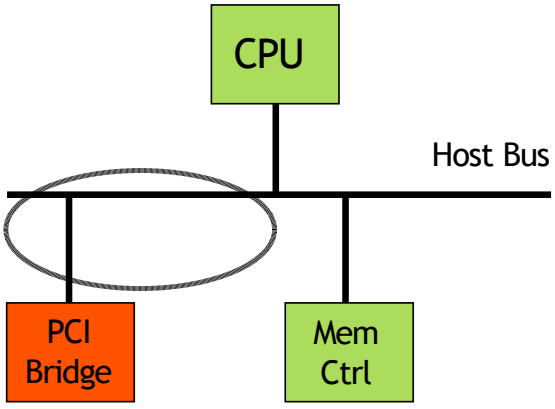

Figure 4a – Partial Host Bus failure.

The branch of the Host Bus leading to the PCI Bridge ASIC could have an open circuit on the line carrying data bit 63 (Figure 4a).

See Application Note #3 Section 2.2 and Section 3.3 for guidance on locating Bus-related faults.

2.3.3.2. PCI Bridge connection Fault.

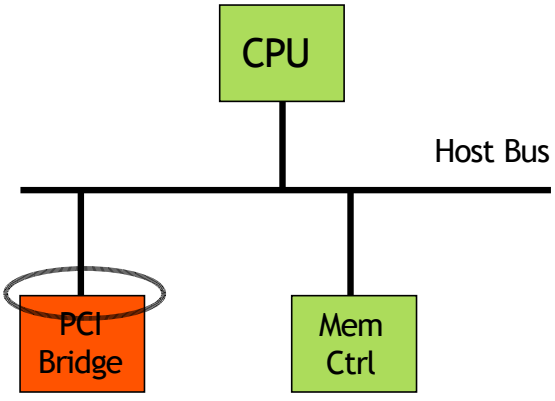

Figure 4b – PCI Bridge connection fault.

The PCI Bridge ASIC's solder joints might not be connected correctly (Figure 4b). This would lead to good information from the Host Bus not getting into the ASIC.

See Application Note #3 Section 2.2 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.3.3.3. PCI Bridge ASIC Failure.

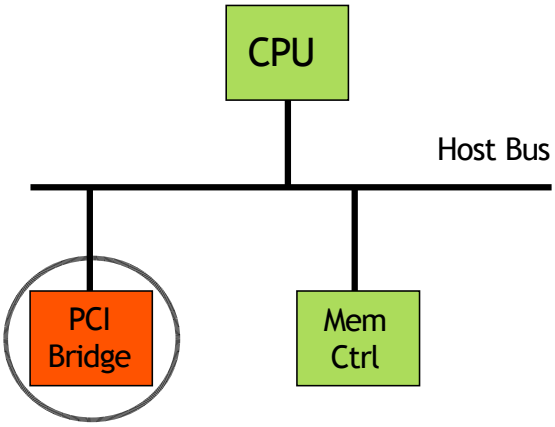

Figure 4c – PCI Bridge ASIC failure.

The PCI Bridge ASIC could be faulty (Figure 4c). The good information that was communicated over the Host Bus has been incorrectly utilized by the ASIC, or has been ignored completely by it.

See Application Note #3 Section 2.3 and Sections 3.1, 3.2, 3.3, and 3.4 for guidance on locating this fault.

2.4. MEMORY FAILURES

2.4.1. Memory Tests

The following tests are used to verify Memory:

- Check access to the Memory Controller registers.
- Set up registers for normal operation.
- Write to memory and then read back to verify.

2.4.2. Example Block Diagram Results

Figure 5 shows that the Memory has failed, but the Memory Controller has passed.

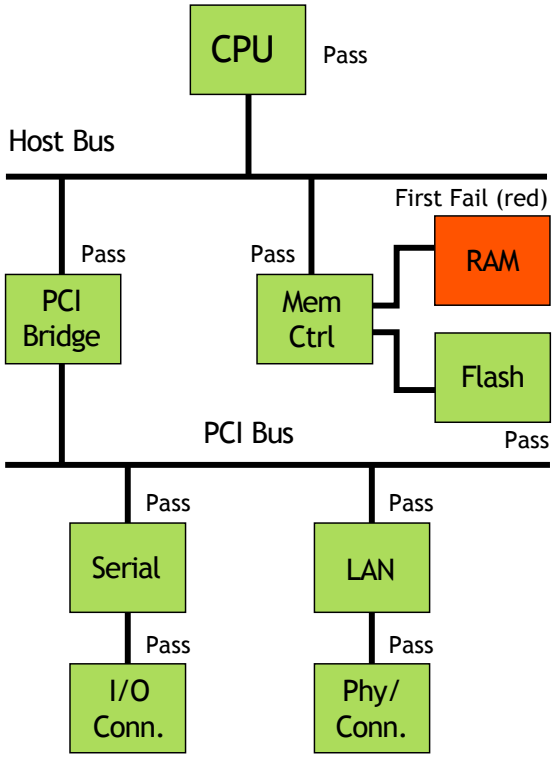

Figure 5 – Memory failure.

2.4.3. Possible Failures and Error Messages

In this case, the failure could lie in part of the back-end of the Memory Controller, the Memory Bus, or the Memory itself.

An example error message would be:

```
Data Bus Stuck High/Low
D31-D16: ++++++++++++++++
D15-D00: +1++++++++++++++
```

This shows that Data bit D14 is stuck high. The following failures could cause this:

2.4.3.1. Memory Controller Back-end Fault.

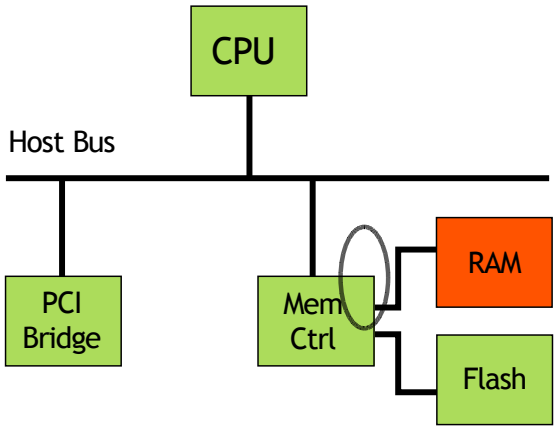

Figure 5a – Memory Controller back-end failure.

Even though the Memory Controller passed its register access and set-up tests and was displayed in green (Figure 5a), the back-end beyond the registers can only be tested by accessing the registers in a child device (e.g. the RAM or Flash). In this case, the Flash tests passed, so this implies that the Memory Controller is good. However, the RAM is on a separate bus so the following faults are possible:

a. Partially faulty Memory Controller: bad data bit D14 to the RAM bus.
b. Bad Memory Controller connectivity. Pin D14 solder joint to the Memory Bus badly connected.

See Application Note #3 Section 2.2 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.4.3.2. Memory Bus Failure.

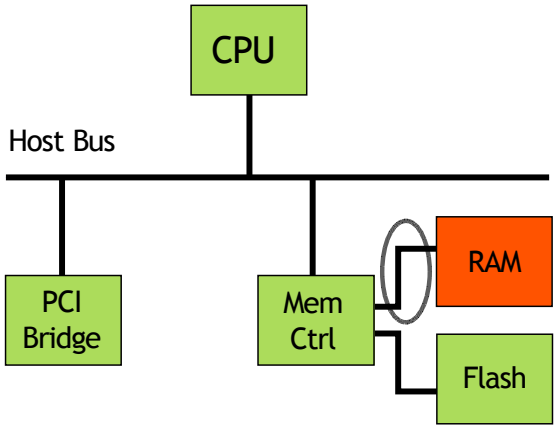

Figure 5b – Memory Bus failure.

If the Memory Controller back-end is not at fault, the cause of the failure could be data line D14 stuck high or short circuited on the PCB's Memory Bus (D14 can't be short circuited to another data line because this would have been indicated in the error message).

See Application Note #3 Section 2.2, 2.4 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.4.3.3. Bad Memory Connection.

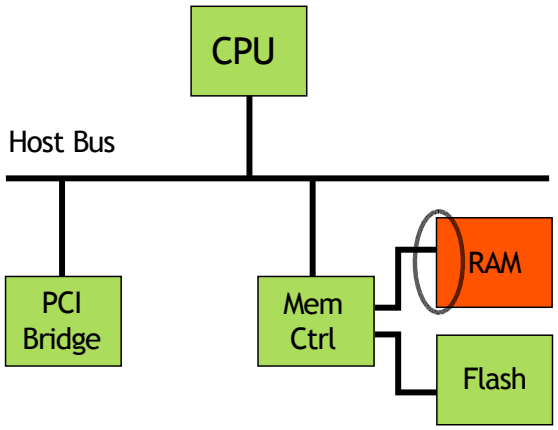

Figure 5c – Bad Memory connection.

If the Memory Controller and Memory Bus lines are good, the cause of the failure (Figure 5c) could be:

a. Badly soldered D14 pin of the Memory or Memory socket onto the Memory Bus.
b. Damaged D14 Memory or Memory socket pin.

See Application Note #3 Section 2.4 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.4.3.4. Faulty Memory.

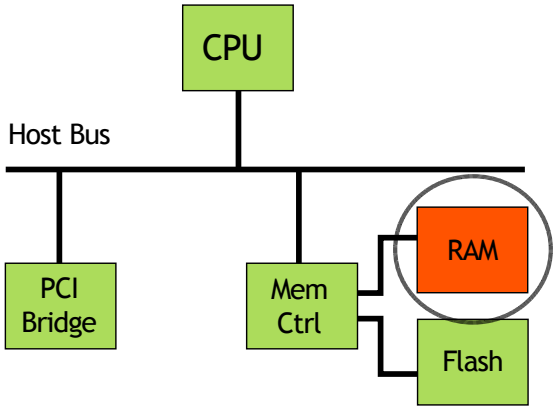

Figure 5d – Faulty Memory.

If the Memory Controller, Memory Bus lines and Memory connections are good, the cause of the failure (Figure 5d) must be faulty Memory.

See Application Note #3 Section 2.4 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.5. I/O CONTROLLER FAILURES

2.5.1. I/O Controller Tests

The following tests are used to verify an I/O Controller (see also section 2.6 for testing I/O Ports):

- Check access to I/O Controller registers.
- Set up registers for normal operation.

2.5.2. Example Block Diagram Results

Figure 6 shows that the Serial Controller has failed. However, the LAN Controller has passed, which implies that the back-end of the PCI Bridge ASIC is good.

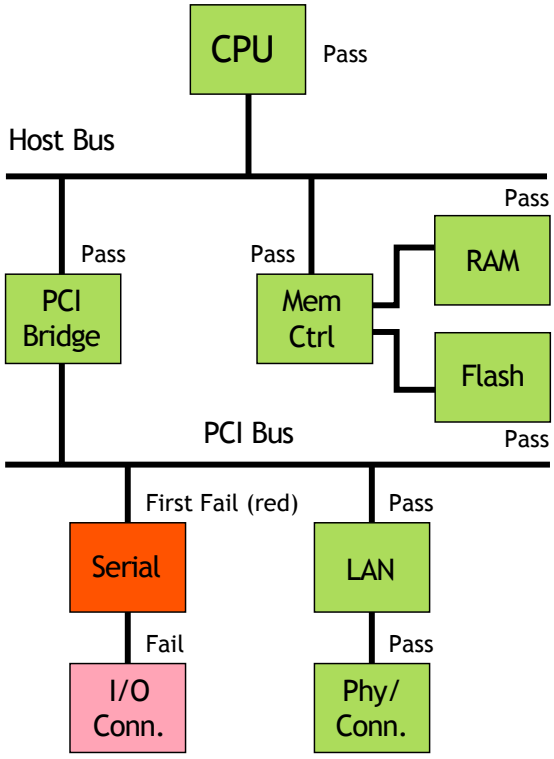

Figure 6 – Serial Controller failure.

2.5.3. Possible Failures and Error Messages

Let's assume that when a register in the Serial Controller was read, the following error message was given:

```
FAILURE Controller Register:
Read     12345678
Expected 02345678
```

If we convert these data values from hexadecimal to binary, we find that data bit 28 is high when it was expected to be low. **It is unlikely to be a stuck high on the PCI Bus because the LAN Controller passed (this is still possible)**. The following failures could be the cause:

2.5.3.1. Partial PCI Bus Failure.

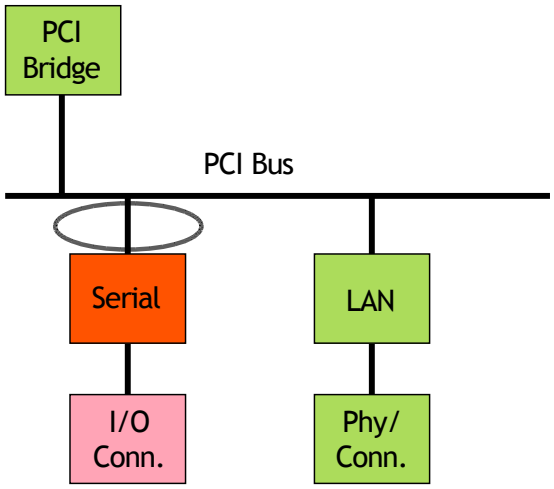

Figure 6a – Partial PCI Bus failure.

The branch of the Host Bus leading to the PCI Bridge ASIC could have an open circuit on the line carrying data bit 28 (Figure 6a).

See Application Note #3 Section 2.2, 2.5 and Section 3.3 for guidance on locating Bus-related faults.

2.5.3.2. Serial Controller Connection Fault.

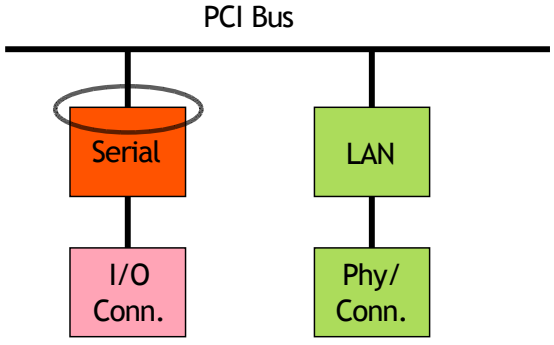

Figure 6b – Serial Controller connection fault.

The Serial Controller's solder joints might not be connected correctly (Figure 6b). This would lead to good information from the PCI Bus not getting into the Serial Controller.

See Application Note #3 Section 2.5 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.5.3.3. Faulty Serial Controller.

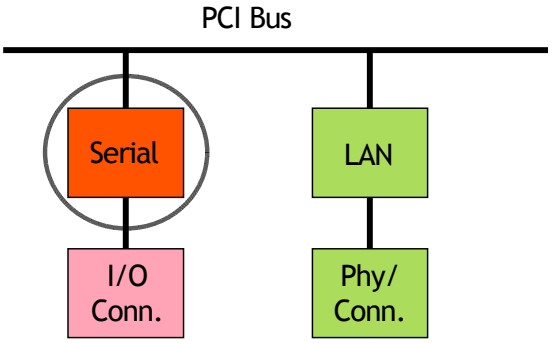

Figure 6c – Faulty Serial Controller.

The Serial Controller could be faulty (Figure 6c). The good information that was communicated over the PCI Bus has been incorrectly utilized by the Serial Controller, or has been ignored completely by it.

See Application Note #3 Section 2.5 and Sections 3.1, 3.2, 3.3, and 3.4 for guidance on locating this fault.

2.6. I/O PORT FAILURES

2.6.1. I/O Port Tests

The following tests are used to verify an I/O Port:

- Having successfully set up the I/O controller registers, transmit data to and receive data from the I/O device or I/O loopback to verify the back-end of the I/O controller and also the I/O Connector.

2.6.2. Example Block Diagram Results

Figure 7 shows that the LAN Port has failed. However, the tests of the LAN Controller's internal registers have passed.

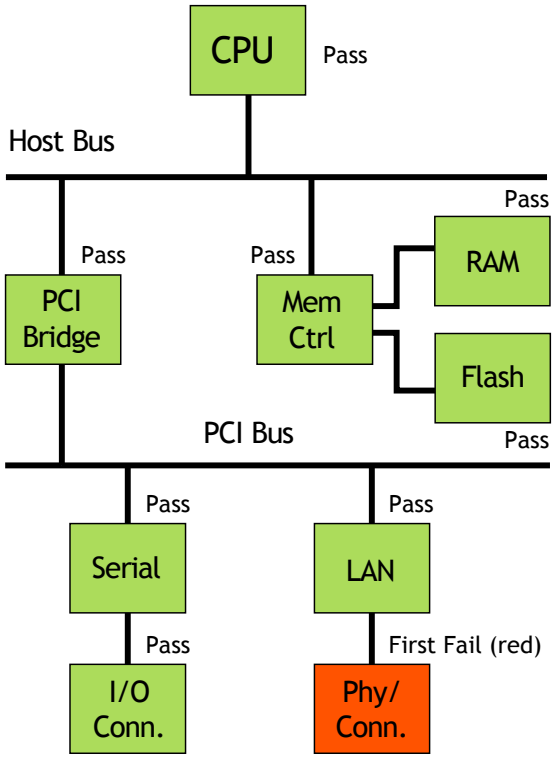

Figure 7 – LAN Port failure.

2.6.3. Possible Failures and Error Messages

A LAN loopback test verifies the connectivity from the LAN controller to the physical LAN port on the target system, right out to the loopback (either a loopback plug or a real LAN cable connected to a switch/hub can be used). It verifies both structural connectivity and also the functional aspects of the LAN controller.

If a read of a memory location failed, the error message would be similar to:

```
FAILURE Loopback Data (10MHz):
Read     00000000
Expected AA559966
```

This shows that the information received was not the same as the transmitted information. The causes could be:

2.6.3.1. LAN Controller Back-end Fault.

Even though the LAN Controller passed its register access and set-up tests and was displayed in green (Fig 7a), the back-end beyond the registers can only be tested by an I/O Port loopback test.

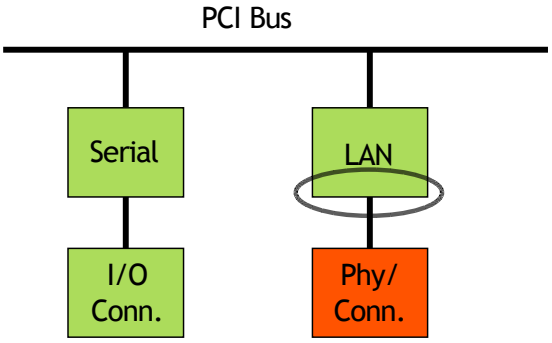

Figure 7a – LAN Controller back-end fault.

The following faults are possible:

a. Faulty LAN Controller back-end.
b. Bad LAN Controller connectivity. The solder joints to the LAN port are not connected correctly.

See Application Note #3 Section 2.5 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.6.3.2. LAN Controller to Port Signal Line Failure.

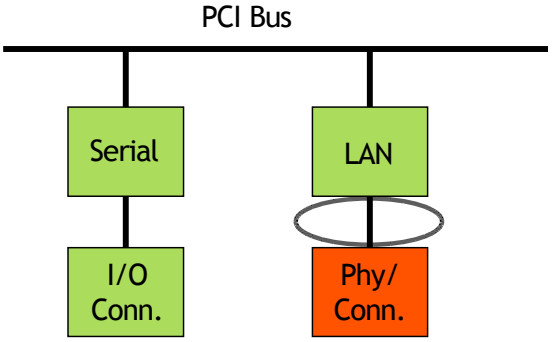

Figure 7b –LAN Controller to Port failure.

If the LAN Controller back-end is not at fault, the cause of the failure could be faulty analogue components or traces connecting the Controller to the Port connector (Figure 7b).

If loopback tests pass at one frequency (e.g. 100MHz) and fail at another (e.g. 10MHz), this is likely to be caused by faulty analogue components, such as capacitors on the LAN bus connecting to the physical LAN connector.

See Application Note #3 Section 2.2, 2.4 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.6.3.3. Faulty LAN Port Connector.

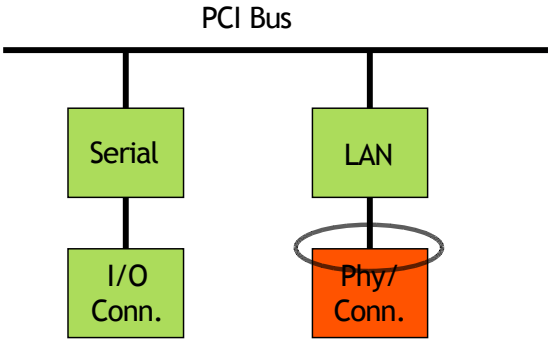

Figure 7c – Faulty LAN Port Connector.

The LAN Port connector's solder joints might not be connected correctly or the connector's pins may be damaged (Figure 7c). This would lead to good information from the LAN Controller not getting out to the LAN device or loopback.

See Application Note #3 Section 2.5 and Sections 3.1, 3.2, and 3.3 for guidance on locating this fault.

2.6.3.4. Faulty LAN Device or Loopback.

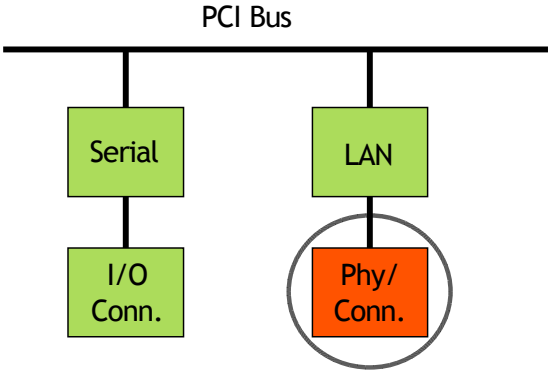

Figure 7d – Faulty LAN Device or Loopback.

Hopefully, a known-good LAN Device or Loopback would be used during a board test, but of course, a failure in this is of course possible.

ASSET CONTACTS:

Please contact your ScanWorks sales representative for more information.

ASSET InterTech, Inc.
2201 N. Central Expy., Ste 105
Richardson, TX 75080
+1 888 694-6250 or +1 972 437-2800
http://www.asset-intertech.com