PCI-SCI Adapter Card D320/D321

Functional

Overview

Version 1.01, November 30 1999

Part no.: D1950-10299

Dolphin Interconnect Solutions
Olaf Heltsets Vei 6
0621 Oslo Norway
Email: info@dolphinics.no
Phone: +47 2316 7000 - Fax: +47 23167180

Dolphin Interconnect Solutions
3609 East Thousand Oaks Blvd, Suite 209
Westllake Village, CA 91362 -3624, USA
Email: info@dolphinics.com
Phone: (805)371-9493- Fax: (805) 371 9785

Table of Contents

1 OVERVIEW

This document is a functional overview of the PCI-SCI Adapter Card (in this document referenced as “PCI Card”) and covers 64-bit version PCI64 (product number D320/D321). The PCI Card is a general purpose high-performance interface to PCI-based systems. The PCI Card use Dolphin’s bridge implementation of the IEEE/ANSI 1596 -. 1992 Scalable Coherent Interface (SCI) protocols to achieve very low latency and high throughput operation . The PCI Card is intended used in clusters of servers and workstations as well as Bus-Bus bridges and I/O expansion.

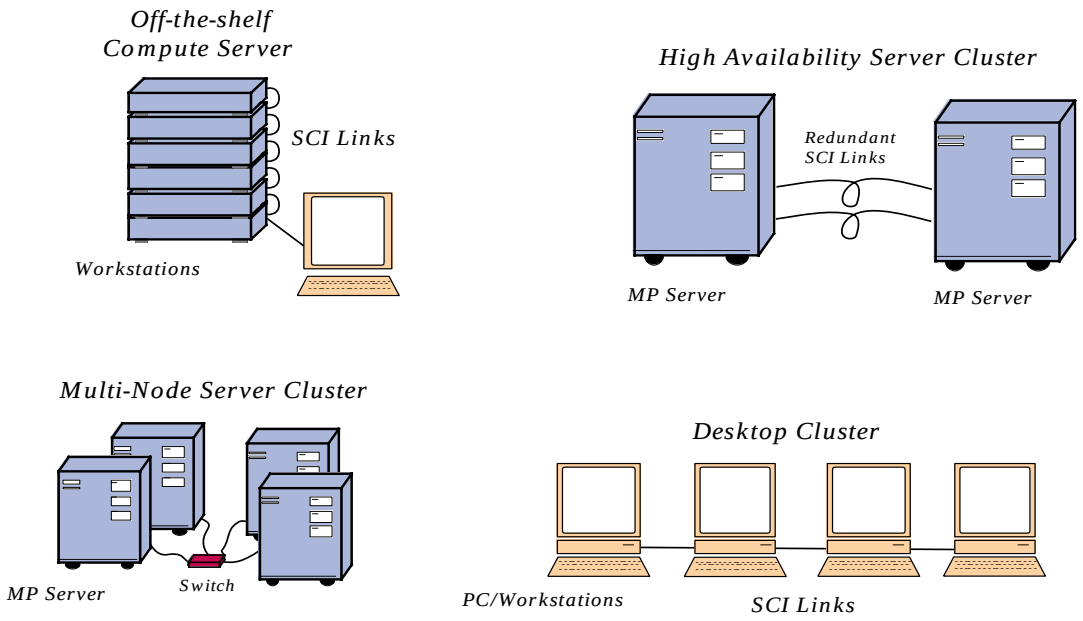


Figure 1. Example system configurations using PCI Card

Applications for PCI Card:

- High Availability database clusters
- Compute server cluster
- Scalable I/O interconnect
- Bus bridging

The basic foundation that makes it possible to do clustering efficiently are:

- Direct mapping of protected remote memory
- Reliable message passing
- Atomic operations
- DMA transfers

Product Features:

- Compliant with PCI specification 2.1. 32-bit/64-bit , 33 MHz PCI bus operation
- Compliant with IEEE/ANSI 1596-1992 Scalable Coherent Interface (SCI) specification
- High throughput (up to 150 Mbytes/sec) read operation
- High throughput (up to 172 Mbytes/sec) write operation

- Low latency memory to memory transfers using load/store
- On-board chained DMA controller to off-load CPU during data transfers (125 Mbytes/sec)
- DMA controller supports both read and write. 4 bytegranularity.
- 400 Mbytes/sec SCI interface with two unidirectional links each 500 Mbytes/sec
- Transparent PCI-PCI bridge operation through memory mapped load/store interface
- Up 64k nodes can be addressed in a system.
- Supports SCI ring and switch topologies.
- Supports SCI automatic configuration
- Optional extra link avaiable on daughterboard (takes one extra slot position)
- Operates with parallel twisted pair cables up to 10 meters between nodes and parallel fiber up to 150 m.

1.1 SUPPORTED SCI PROTOCOLS

The PCI-SCI bridge supports a compatible and compliant subset of the full SCI protocols. The PCI-SCI bridge does not support the SCI hardware cache coherence protocols. Maintaining cache coherence between applications must be solved by using non-cached operation or caching with explicit software cache coherence maintenance. The following SCI protocols are supported by the PCI Card:

- Non-coherent 64-byte read and write (Nread64, Nwrite64, Dmove64)
- Non-coherent 1-16-byte read and write (Readsb, Writesb, Movesb)
- SCI locks (Locksb). All 4 bytes types except LITTLE_ADD handled as master.
  Slave can generate any Locksb (or any other packet type) using the SW packet interface.
- PCI64 supports up to 64 outstanding SCI transactions (32 read/32 write)
- Supports 2 outstanding SCI subactions
- 18 bits parallel transmit and receive SCI links
- Hardware busy retry protocols
- 16-bit SCI packet CRC code
- Fair arbitration on SCI ringlet
- Supports max SCI packet size of 80 bytes

1.2 SUPPORTED PCI COMMANDS

The PCI Card supports the following PCI commands as master and slave if not otherwise noted:

- I/O Read and Write (master only)
- Memory Read and Memory Write
- Configuration Read and Configuration Write
- Memory Read Multiple (as DMA master)
- Dual Address Cycle (slave only) - uses 64 bit PCI address as SCI address.
- Memory Read Line
- Memory Write and Invalidate (as master, cache-line sizes 0,2,4,8,16 are supported)

2 FUNCTIONAL DESCRIPTION

2.1 BASIC OPERATION

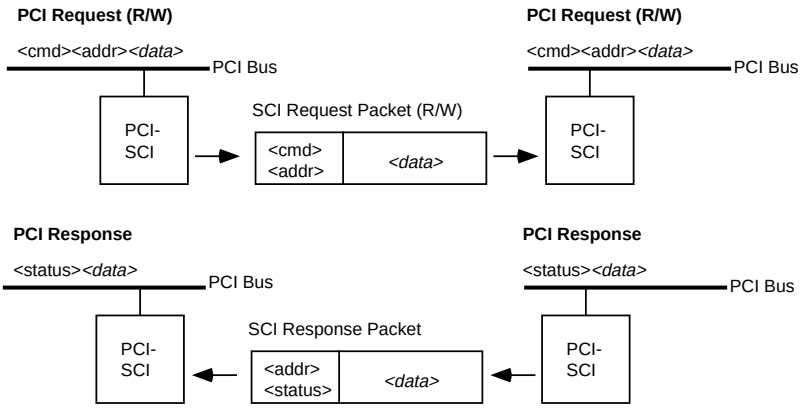


Figure 2. Basic PCI-SCI bridge operation

A PCI master can read and write memory and registers on another PCI bus via the SCI interconnect. The PCI-SCI Card also includes a DMA controller that allows the PCI-SCI Card to transfer data between PCI buses.There are basicly two types of operation, Programmed I/O(PIO) or DMA .

2.1.1 PIO Operation

PIO operations are singel read or write requests towards the PCI-SCI bridge The PCI-SCI bridge transparently forwards requests and responses between PCI buses. A PCI read or write request on one PCI bus is mapped into an SCI read/write request and sent to the target PCI bus. The target PCI bus will execute the read/write request and respond accordingly. The response is mapped into an SCI response packet and sent to the PCI bus where the request originated. Figure 2.

The transparency between buses is achieved by PCI-SCI command mapping and address translation. A controller chip on the PCI-SCI Card maps PCI requests into SCI request packets. An on-chip address translation unit maps the PCI bus address into a 64-bit SCI address. This address is used to address the target node and the PCI device residing on the target node. A node can have many PCI buses and PCI devices. The PCI-SCI Card supports several addressing modes capable of addressing from 256 to 64k nodes in a single system.

For high bandwidth applications, the PCI-SCI Card use write posting to achieve high throughput operation on write streams. SCI packets can be pipelined to achieve data streaming performance close to the maximum write performance on a PCI bus. For smaller PCI data transfers, write gathering can be enabled to increase performance. High read throughput performance is achieved using read prefetching coupled with use of delayed PCI transactions. For read or write to consecutive addresses, the PCI-SCI adapter could give a memory performance similar to a local PCI memory

Write before read ordering will be enforced when the Ordering attribute inthe Address Translation Any read request will stall the packet pipeline until all outstanding write requests are done.

2.1.2 DMA operation

DMA operations will use the on chip DMA engine. When started this engine will execute from an already build execution queue in the hosts memory. This means that the PCI-SCI bridge now is in complete control of the operation. Data is requested from the memory and pushed onto SCI terminated by the word count in the control block. DMA operations are chained in the execution queue, and when queue is empty, completion is signalled.

2.1.3 Interrupts

Interrupts can be sent over SCI by explicit device driver operation. It is also possible to transparently forward an interrupt from one PCI bus to another PCI bus. Executing a lock read-modify-write operation on a different PCI bus is possible by explicit device driver operation that generates an SCI lock request that becomes a normal lock operation on the target PCI bus.

2.1.4 Reliability

Reliable operation is achieved by the SCI flow control protocols. SCI defines guaranteed data delivery through a set of request/response protocols coupled with buffer to buffer flow control. In the case of filled buffers, SCI link interface chips will perform busy retry in hardware and guaranteed delivery. In the event of errors, SCI response packets will report error status to the PCI interface (configurable interrupt or SERR#). Missing responses are detected by programmable time-out counters. Write ordering and delivery can be controlled by store barrier operations supported by the PCI-SCI Card.

2.2 PCI-SCI CARD BLOCK DIAGRAM DESCRIPTION

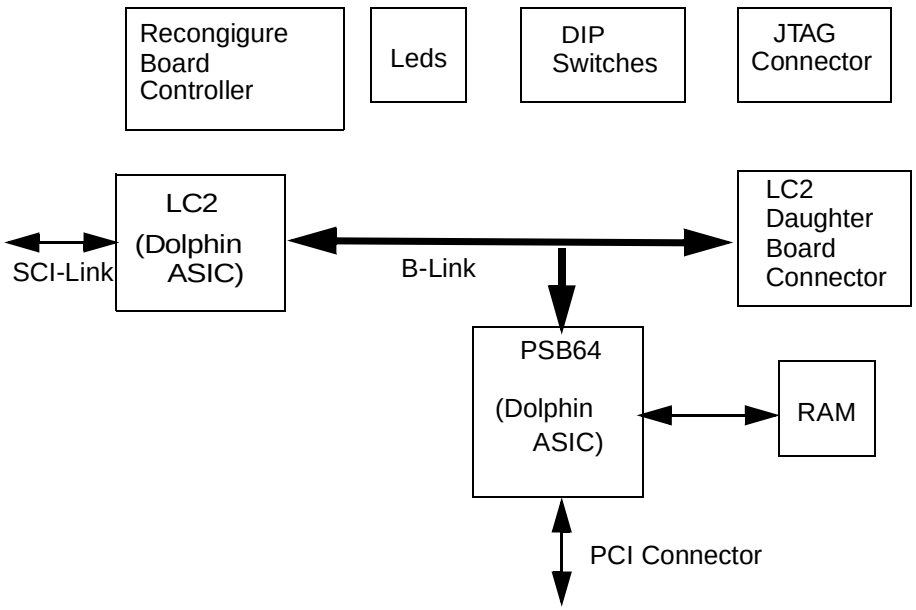


Figure 3. PCI-SCI Card D320/D321 Functional Block Diagram

The PCI-SCI Card is focused around the B-Link bus. It operates as a backbone link between the Link Controller (LC2)chip and the PSB64 chip. The blocks are described below.

2.2.1 PSB64

The PSB64 is a bridge between the PCI bus and the B-Link bus and converts PCI bus memory transactions into SCI bus memory transactions, and vice versa. It transfers data between the buses in both directions. It supports both 32 and 64 bit SCI addresses.

2.2.2 LC2

LC2 is the Link Controller on the PCI-SCI Card. It manages data transfers on the SCI physical level by sending and receiving packets on external SCI links. LC2 also interfaces the B-Link and uses this for transferring data between the external interfaces and the PCI bus (via PSB64). LC2 includes routing functionality that determines if the incoming data is destined for “this” node. If not, the data is placed in a bypass FIFO which forwards the data directly to the output link.

2.2.3 Board Controller

The PCI-SCI Card contains a Reconfigure Board Controller (BOARD_CTRL) consisting of two PLDs (PLD_A and PLD_B). Each can be reconfigured in the field (ISP = In System Programming) via their JTAG ports or by SW-download. The Controller controls reset, I2C initialization of the Link Controller(LC), board status reporting, and LED function. Via CSR accesses to the PSB the driver SW can read and write control- and status registers in the Controller.

2.2.4 DIP Switches

There are 10 DIP switches on the PCI-SCI Card, that enable the JTAG programming and production test of the card, set the SCI link frequency to 100MHz or 125MHz, set the Blink frequency to 50MHz or 66MHz, set the Slave windows on PCI to 16MB, 64MB, 256MB or 2GB and accept/ignorer 64 bits addressing as PCI slave.

2.2.5 Daughter Card Connector

The Daughter Card Connector may be used to connect a LC2 Daughter Card to provide fault tolerance and increased routing ability. The Daughter Card has external SCI links.

2.2.6 B-link activity LED

If yellow: cable error or sync error

If green steady: operational, no B-link transactions

if green blinking: operational, B-link transactions going on

2.2.7 SRAM

The SRAM chip is used by the PSB64 for storing Address Translation Tables.

2.3 PSB64 AND LC2 DESCRIPTION

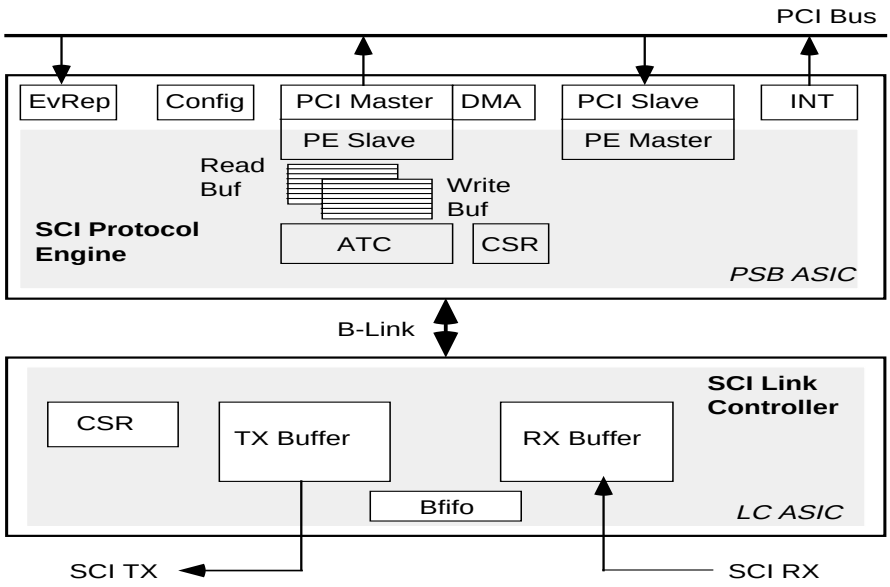


Figure 4. Block diagram of PCI-SCI bridge

Figure 4 illustrates a block diagram of the PCI-SCI connection. The bridge is implemented in two ASICs. A PCI to SCI bridge (PSB64) chip implements the protocol mapping between PCI and SCI and vice versa. An SCI Link Controller (LC2) chip implements the SCI transport layer. The PSB chip interfaces to the PCI bus and maps PCI commands and address to SCI transactions through the SCI protocol engine. The protocol engine generates the SCI packet formats and sends the packets to the SCI Link Controller chip. The LC2 chip is responsible for delivering the SCI packets to the target node and guarantee delivery through the SCI flow control protocol.

2.3.1 PCI Interface

The PCI interface of the PSB chip operates according to the PCI specification ver. 2.1. It contains the PCI master and slave interfaces as well as PCI configuration registers. PCI devices can receive interrupts triggered by the protocol engine as a result of errors or interrupt register operations. The PCI interface contains a DMA controller for high performance memory to memory transfers.

2.3.2 Read and Write Buffers

A transaction gets assigned a stream consisting of a date buffer, an ATC entry (see below) and status bits.

The protocol engine contains 16 streams for read and 16 streams for write operations, each with a128 byte buffer. These buffers are used to support posted PCI write transactions and read prefetch transactions.
PCI to PSB CSR space transactions are routed through separate CSR read and write streams, one of each.

2.3.3 Address Translation Cache (ATC)

The address translation cache (ATC) holds address translation entries. The table is used when the protocol engine needs to map a 32-bit PCI address into a 64-bit SCI address. The tables also contain information relevant to the address being accessed (page attributes). The PSB64 internal ATC has one entry per stream for holding the most recently used address translation table entries for the different streams. The complete address translationtable (ATT) is located in an on-card SRAM. An ATC miss will cause the PSB to fetch a new table entry from this SRAM.

2.3.4 Control and Status Registers

The PSB and LC chips both contain a number of CSR registers. These registers are used for error logging, initialization, status information etc. CSR registers are accessible both from PCI and SCI.

2.3.5 DMA

The PSB64 chip contains a high-performance DMA controller (DMA Engine). This DMA controller can be used as an alternative memory-memory transfer method to CPU load/store and device DMA. The DMA controller utilize the most efficient SCI transactions and PCI commands to achieve high throughput operation.

2.3.6 Protection

The PSB Master has a two level mechanism to protect the PCI bus and the CSR registers from errant hardware/software accesses. It consists of a common mechanism that protects the PSB from requests from specified source nodes, an address protection for accesses to the PCI bus and a special CSR access protection.

Note: There is NO outbound (i.e. SCI/B-Link request) protection/checking.

2.3.7 SCI RX and TX Buffers

The LC chip contains buffers for temporarily storing SCI packets that are being transmitted and received. Both the TX and RX buffers have separate queues for request and response packets to guarantee deadlock free operation according to the SCI specification.

2.3.8 SCI Links

The PCI-SCI Card has two unidirectional SCI links that connect nodes in a system. Each link has 16-bit wide data lines and a control flag and clock signal. All signals are differential and can connect directly to copper cables without additional transceiver circuitry.

3 PCI AND SCI COMMAND MAPPINGS

3.1 STREAMS AND BUFFER ASSIGNMENT

The PCI Card use the concept of streams to support several simultaneous PCI masters and resource allocation of the read and write buffers. There are 16 write streams (128Bytes) and 16 read streams (128Bytes). A direct mapping from PCI addresses to buffers is used. The effect of this mapping can be changed by configuration to combine streams.

Basically the stream selection mechanism works like this: A four bit value is selected from the upper four bits of the address window. When no stream combining is configured, this value selects the buffer (0-15). Stream combining is done by replacing some of the lower bits of the four-bit value with the lower address bits 7-10 depending upon the level of combining (1, 2, 4, 8 or 16 buffers).

Stream combining means higher throughput performance. The level of combining is selected through a CSR register.

3.2 WRITE FROM PCI DEVICE ACROSS SCI

Common to all PCI writes to the PCI-SCI bridge is that they are all posted writes. In normal operating mode, a PCI write will be mapped into an SCI write transaction. PCI burst writes of less than 64 bytes will be mapped into one or more selected-byte SCI write transactions. PCI writes of 64 bytes will be mapped directly into a 64-byte SCI write transaction.

PCI write size	SCI command
1-63 bytes	n * Write_SB
64 bytes	n * Nwrite64
> 64 bytes	n * Nwrite64, n * Write_SB

The PSB also supports 128 byte packets. However since the LC2 only supports 64 byte SCI packets, a PCI burst write of 128 bytes are accepted by the PCI Card, but will appear as two 64 byte SCI transactions.

The PCI slave interface will do a retry when a 128 byte boundary is crossed. In order to generate the 64 byte SCI write packet, the start address to the buffer must be 64 byte aligned.

3.2.1 Write Gathering

The write gathering function is important to use when the master device is not capable of transferring 64 bytes or more in a burst. In write gathering mode PCI data are gathered until the 128 byte boundary is crossed, whereupon the buffer is automatically flushed (written across SCI). The number of SCI transactions generated depends on the start address of the gathered data. A buffer may also be explicitly flushed by a register write, by an interrupt detected by the event reporter (if enabled), or by a buffer time-out. Write gathering ensures high throughput operation even when a master transfers small amounts of data since the efficient 64 byte SCI transactions are used. Write gathering is enabled through a PCI configuration register and can be selected per page. The gathering granularity is 4 byte, i.e. a PCI write of 1, 2, or 3 bytes, or a PCI write with non-consecutive byte enables (byte holes) will enforce a buffer flush.

3.2.1.1 Flush Buffer

When doing write gathering, flushing will be used to send off partially filled buffers. The flushing procedure may be initiated by a read or write to the SW_FLUSH register, by a read of the PAGE_BARRIER register, by an ordered read or ordered write, by an external interrupt (INTA/B/C/D), or by a buffer timeout. An ordered request means it will ensure that flushed data have been delivered to the destination node, i.e. SCI response received, before the ordered request itself is sent on to SCI. The status of the flushing can be monitored by reading the SL_DEADLK_CNT register.

A buffer will also be flushed if the buffer is demanded by a new PCI request, i.e. an address not expected by the gathering. The buffer will be flushed, and the new request will get the buffer when the flushing is completed.

3.2.2 DMA write transfers

The PCI Card contains a DMA controller that can be used for memory to memory transfers across SCI. Instead of using the CPU to move data or use a PCI device's DMA controller, driver software can set up the PCI Card DMA controller to transfer data. The DMA controller will read data from local PCI memory and write data across SCI to remote PCI memory (DMA push) using 64 byte SCI packets. The DMA controller supports chaining.

3.2.3 Store Barrier

A store barrier can be used when software needs to know when all outstanding write operations are finished. By writing to the STORE_BARRIER register, a snapshot of all the write streams with status Posted (written to SCI and waiting for response) is taken. When the status change to Free, the corresponding bit in the STORE_BARRIER is cleared. By polling this register software can determine when all outstanding writes have completed. Note that new outstanding writes coming after the setting of the last store barrier is not reflected in the register.

A store barrier can also be achieved by issuing a PCI read command to a CSR register space of 4k (PAGE_BARRIER) on the PCI Card. Reading a PAGE_BARRIER entry will flush all streams using this page.

3.3 READ FROM PCI DEVICE ACROSS SCI

The PCI-SCI interface support two basic read mechanisms. An aggressive prefetch can be used by high bandwidth applications, while a direct mapped mechanism is used to read I/O registers or used by applications that do not require high bandwidth operation.

PCI Space + config	SCI commands
Mem I/O	Read_SB
Mem Prefetchable	Nread64
Mem Prefetchable + aggressive	n * Nread64

These two methods are selected by addressing either the prefetchable memory range or memory IO range PCI address spaces of the card. The sizes of these address spaces are configurable from 16 Mbytes to 2Gbytes.

All reading via the PCI Card is based on retry on PCI (except configuration cycles) due to long latency on SCI compared to the PCI bus. Even reading internal CSR registers will use retried transactions.

3.3.1 Memory I/O Read

When accessing the memory I/O address space only read selected byte of 4 bytes are requested. Any burst attempt will be retried forcing a new 4 selected byte read access.

3.3.2 Memory Prefetch Read

The read prefetch mechanism may be enabled perpage by a bit in the ATT entries. 64 bytes (non-aggressive prefetch) or 128 bytes (aggressive prefetch) are prefetched. In order to really get the speed up on read transfers the aggressive prefetch mechanism can be enabled by another bit in the ATT entries. Using aggressive prefetch mode together with stream combining will give maximum performance. Aggressive prefetch will automatically re-prefetch a buffer when it has been emptied, taking the actual stream combining into account when calculating the new prefetch address. The PCI Card will thereby attempt to keep all stream buffers full at any time so that subsequent PCI reads always get the data from the PCI Card without waiting for an SCI read transaction.

Depending on the number of streams combined the PCI Card will send multiple 64 byte SCI read transactions. If all 16 streams are combined and the aggressive prefetch bit is on in the ATT entries, a PCI read will generate as many as thirty-two 64 byte SCI transactions to fill all the buffers.

3.3.3 DMA read transfers

The PCI Card has a DMA controller that can be used to pull data (DMA pull) into a node by setting up a DMA transfer (DMA push: ref section 3.2.2). When reading the DMA controller will generate a number of 64 byte SCI read requests and send them to remote node, which will read the data from its local PCI memory and respond back with 64 byte responses.

3.4 INTERRUPTS & MESSAGES

The PCI Card has one interrupt output. This output is connected to the INTA# PCI interrupt signal. The interrupt can be asserted in different ways.

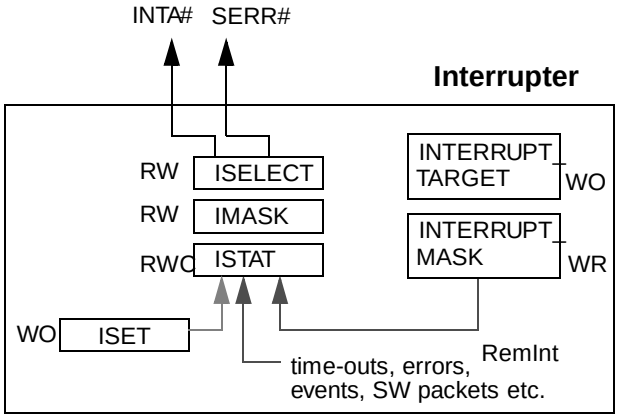


Figure 5. Interrupter

The basic interrupt flow in the interrupter can be se en in Figure5. The boxes represents CSR registers.

3.4.1 Card Internal Events

Different error situations and completion conditions can be programmed to give INTA# or SERR#. This include e.g. packet time-out, DMA completion, Master Abort detected etc.

3.4.2 CSR generated interrupt

A device driver can send an interrupt across SCI by writing to the interrupt target register on the PCI Card that should assert the INTA signal.

3.4.3 Message or SW packet received interrupt

A device driver can send an interrupt across SCI by writing an SCI packet to the odd nodeID of the remote card. The receiving PCI Card will decode the address and write the whole SCI packet, called Software packet, to a memory ring-buffer and at the same time optionally assert the INTA signal.

This Software packet feature can be used when simulating SCI logical protocols (including cache coherence) and an application can use the PCI Card as a transport mechanism for SCI packets.

The main use of this ring buffer is for implementing the message passing protocol in the device driver. See section 3.7.

3.5 TRANSPARENT INTERRUPT FORWARDING

The PCI Card has also four input interrupt signals INTA, INTB, INTC and INTD. Asserting an input interrupt signal can cause the interrupt signal on a different PCI bus to be asserted. Setup registers on the PCI Card define where interrupts are to be routed and which interrupt method to use.

3.6 ATOMIC OPERATIONS

A device driver on one system can generate a read-modify-write atomic operation on another remote system using the SCI lock selected byte transactions. The SCI lock packets can be generated in two ways:

The atomic operation is triggered by a device driver writing lock arguments into memory. The driver triggers the PCI Card to read the arguments and generate an SCI lock. The SCI lock packet is received by the remote PCI Card, and the read-modify-write operation is carried out. Upon completion the remote PCI Card sends a response packet, which is written to memory (SW response packet) and an interrupt triggers the device driver to read the results of the remote lock operation.

The atomic operation is triggered by accessing a page address that has the Lock bit set in its ATT entry. Accessing such a page will result in the generation of a lock selected byte operation FETCH & ADD +1. Read and writes to this page will generate the same packet except that on the write version a remote interrupt (lock interrupt) will be generated if the old value was zero. This is referred to as a conditional interrupt. On reads the response packet contains the old value which is returned to the PCI device accessing the page.

As a master the PCI Card supports all 4 byte lock operations (except LITTLE_ADD) defined in the SCI specification. A software packet must be used to trigger the following lock operations:

4 byte MASK_SWAP
new = (data&arg) | (old&~arg)

4 byte FETCH_ADD
new = old + data

4 byte BOUNDED_ADD
if (old != arg) new= data + old ; else new = old

4 byte COMPARE_SWAP
if (old == arg) new = data; else new = old

4 byte WRAP_ADD
if (old != arg) new = data+old; else new = data

3.7 SW PACKET INTERFACE

SW packets are just like other SCI packets only that an odd node ID is used as target ID. The PCI Card will handle these request packets differently. The PCI Card will accept SCI requests with both the even (normal packets) and the odd node ID. The card using this interface can both send and accept all types of SCI packets. The packet decoding and building must be done with software instead of hardware as with normal packets. Coherent nodes can in this way be emulated in SW and message passing protocols can be implemented.

The destination of a SW request packet can both be a normal or a SW packet destination, but the source ID of a request must be the odd node ID of the PCI Card. Otherwise the response packet might corrupt a response buffer.

This interface can be used to implement higher level SCI protocols in software using only the SCI link layer for packet transportation.

The interface can also be used as a method for reliable message passing. The receive side of the interface can be configured to automatically generate responses on requests. The message sender will get immediate info that the message is stored in the receiver.

3.7.1 SW node receive ring buffer

SW node operation means that a raw SCI packet (it is encapsulated in an PSB internal format packet which is similar to a B-Link packet minus the last beat) is to be transferred to a PCI memory ring buffer.

All request packet entering the PSB with the odd target node ID will be directed into the ring buffer. Packets not passing the source ID check will not be stored in the buffer and a response packet with response type RESP_ADDRESS will be generated (if the request command expects a response).

If the ring buffer is full, a RESP_CONFLICT is returned (if the request expects a response).

The size of each buffer entry is 128 bytes and the ring buffer is maintained by the following registers:

SW_Q_START/SW_Q_END: Defines the size of the ring buffer.

SW_Q_HEAD: Maintained by PSB

SW_Q_TAIL: Maintained by Software that reads the ring buffer.

The buffer full condition is: Expected Next SW_Q_HEAD ==SW_Q_TAIL.

When a SW packet is received the ISTAT.SwPacket bit is set for interrupt generation.

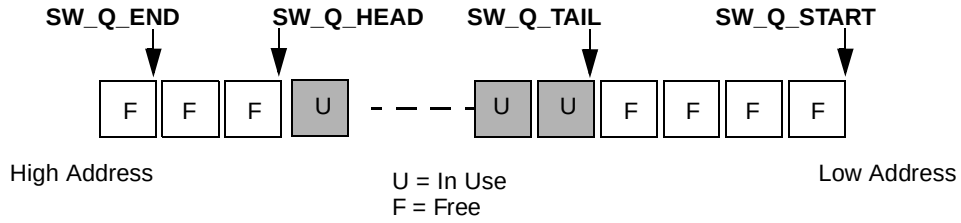


Figure 6. SW packet ring buffer

3.7.2 SW Packet send

A B-Link packet (an encapsulated SCI packet) is stored in memory and pointed to by PKT_POINTER register. The number of required beats for the stored packet is written to PKT_SEND register and the packet is sent into SCI. The status of the transfer can be checked by reading PKT_SEND.

3.8 DMA ENGINE

3.8.1 Overview

The DMA Engine requests both the PSB master and the PSB slave when running, but other PCI masters will also get access to PSB slave between each packet transferred (maximum 64 bytes). Also traffic from SCI will get access to the PCI bus when the DMA Engine is running. Only a lower performance in PSB is seen from PCI and SCI side by other PCI masters.

3.8.2 General features:

- The DMA engine operates with word (4 byte) granularity.
- Can use both 32 bit addressing (using ATT table) or 64 bit addressing (using a CSR).
- Block transfer size is from 4 bytes to 256 Kbytes.
- Programmable interrupts after finishing of jobs.
- Both DMA write (push) and DMA read (pull).
- Gives interrupt and stops on any error.
- The DMA engine can operate in two modes: single mode or chained mode.
- Can be used in loop-back to move data in PCI memory efficiently.

3.8.3 General

The DMA control block specifies a Window Select (Winsel) value and can therefore use any stream regardless the address and can also use combined streams for higher bandwidth.

DMA Status:

The status of the DMA engine can be monitored by reading the DMA_STATUS register. The DMA engine can be programmed to issue an interrupt after finishing a control block, at the end of chain or when an error occurred (See ISTAT.DmaInt).

Effective Use:

When reading from SCI the PSB will use the PCI memory write commands to move data into PCI memory. If both the PCI address and the SCI address are 64 byte aligned, 64 byte SCI packets can be used, giving highest possible performance.

3.8.4 Single DMA Mode

In single mode the registers must be set up by writing to the DMA CSR registers to specify the PCI address, SCI address, word count, destination and DMA engine control. Refer the PSB register descriptions.

The procedure of setting up a single DMA transfer is:

1. Set DMA_SCIADDR
   A mapping to this address must exist if 32 bit addressing is used. If 64 bit addressing mode is used, the DMA64_BASE must also be set.
2. Set PCI_ADDR
   An address in PCI address space (not PSB range!)
3. Set DMA_CTRL
   Word count, direction, buffer window selection, addressing mode.
4. Set DMA_CMD
   Enable DMA channel. No Chaining.

3.8.5 Chained DMA Mode

In chained mode the different DMA jobs are set up in memory and loaded automatically by the DMA controller. The DMA jobs are specified by control blocks that contains the contents of the DMA registers for each job. The control block has the following format:

0h	DMA_PCIADDR
4h	DMA_SCIADDR
8h	DMA_CTRL
ch	DMA_64_BASE

The control blocks are located in PCI memory space in a sequence with increasing addresses (+16 bytes). The first control block is pointed to by the DMA_LPTR register. The last control block must have the EOC (end of chain) bit set in the control word. The chaining DMA setup procedure is:

1. Set up the DMA_LPTR to point to the first control block.
   The control blocks are stored in memory.
2. Write to DMA_CMD
   Enable chaining and DMA channel.

4 ADDRESS TRANSLATION AND PROTECTION

The PCI Card accepts both 32 bit and 64 bit PCI addressing. A 64 bit PCI address is only slightly modified (see figure). A 32 bit PCI address is translated into a 64-bit SCI address using an address translation table(ATT) residing in an SRAM on the card. Address translation table entries are also called Page Descriptors and are cached in the PSB address translationcache. The ATT has 4k entries of 64 bit each.

For incoming SCI requests address protection can be used to disable access to the node or certain PCI memory regions.

4.1 OUTGOING PCI-SCI ADDRESS TRANSLATION

In 32 bit addressing 4 PCI slave window sizes are implemented: 16M, 64M, 256M and 2G bytes. This is reflected in how many SCI offset bits are copied from the lower PCI address bits. The remaning offset bits as well as the SCI node ID are taken from the ATT entry. The mapping is illustrated in Figure 7.

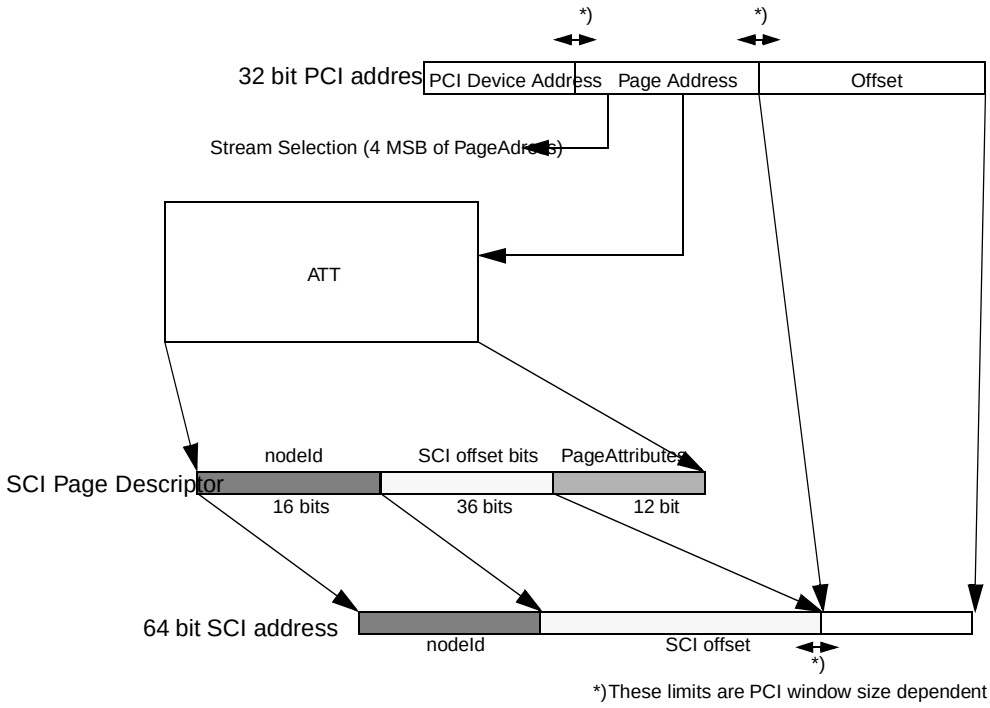


Figure 7. 32 bit PCI -> SCI address

4.1.1 Address Translation Cache

Table entries are stored in a 32-entry address translation cache in the PSB chip, one entry for each stream. A PCI address is checked against the cache. If a miss isdetected, an ATT lookup is performed.

SCI Page Descriptor Layout

Each SCI Page Descriptor (SPD) is 64 bit wide and contains the following fields (also refer Figure 7.):

NodeID - ATT[63:48]. Selects the SCI nodeID of the destination.

Offset - ATT[47:12]. The upper 36/34/32/29 bits of the SCI address sent. The lower 12/14/16/19 bits are taken directly from the PCI address (based on the PCI window size and ATT page size selected in PCIMEMSIZE).

Prefetch - ATT [11]. When set, the PSB64 will prefetch 64 bytes (Aggr 0 0) or 128 bytes (Addr 0 1). When cleared, the PSB64 will only generate 1-8 byte reads and writes, which are intended for reading and writing of registers (either in IO or memory space).**Read** - ATT[10]. When set, reads are allowed to the SCI page.

Write - ATT[9]. When set, writes are allowed to the SCI page.

Lock - ATT[8]. On read, send a **locksb** packet with subcommand, FETCH_ADD with argument 1 will be sent. On write address bit 47 is set in addition.

Ordering - ATT[7]. Write before read.

Dmove -ATT[6]. Use SCI directed move (responseless).

Gather - ATT[5]. Enable write gathering.

Hold - ATT[4]. Enable speculative buffering (on reads).

Aggr - ATT[3]. Enable aggressive prefetch, 1.e. prefetch 128 bytes if prefetch is set.

Use128 - ATT[2]. Enable use of special 128 byte SCI packets (chopped nread256/nwrite256). Only possible with LC-3 and PSB64.

Resv - ATT[1]. Reserved. Must be written with zero, and ignored on reads.

Parity - ATT[0]. Odd parity. Generated by hardware on writes, and checked on reads. If ATT parity is disabled (GIU_CONTROL.AttParEn), this bit is taken from the data written, i.e.: it is software controlled (for test purposees)

4.1.2 64 bit mode

To avoid using the whole 64 bit address space on PCI, the PCI address is slightly modified to get the 64 bit SCI address, as shown in Figure 8.

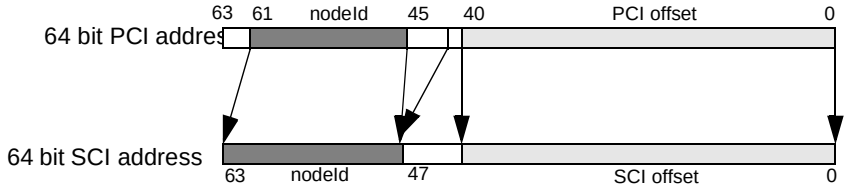


Figure 8. 64 bit PCI -> SCI address

NodeID - PCI[61:46]. Selects the SCI nodeID of the destination.

Window Select - PCI[45:42]. Select the stream for the transfer.

CSR Select - PCI[41]. Selects CSR addressing by forcing the SCI address bits 47:41 to one. If zero, the SCI address bits 47:41 are forced to zero.

Offset - PCI[40:0]. The offset bits taken directly from the PCI address.

Write before read ordering in 64 bit addressing mode is enabled by GIU_CONTROL.Order64bitEn. The ordering is enabled for all write streams.

The registers ATT_PIO64 (143) and ATT_DMA64 (141) sets the transfer attributes for PIO and DMA since there is noATT entry for 64 bit addressing.

Note: Locking is not supported in 64 bit mode.

4.1.3 Local CSR mapping

The PCI Card has a 32 Kbyte address space for accessing CSR registers.

.

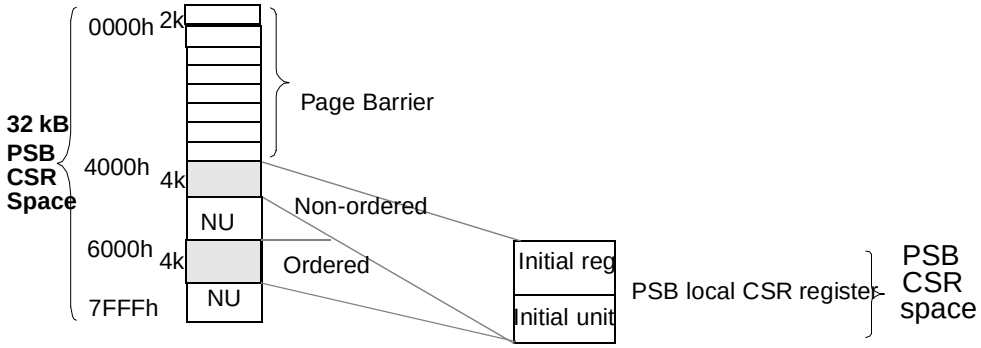


Figure 9. PCI Card Register space mapping

4.2 INCOMING REQUEST PROTECTION

The PSB has a two level mechanism to protect the PCI bus and the CSR registers from unwanted accesses. The first level consists of a common mechanism that protects the PSB from requests from specified source nodes. The second level consists of an PCI bus access protection and a CSR access protection respectively. Refer Figure 10.

Note: There is no outbound (i.e. SCI/B-Link request) protection/checking.

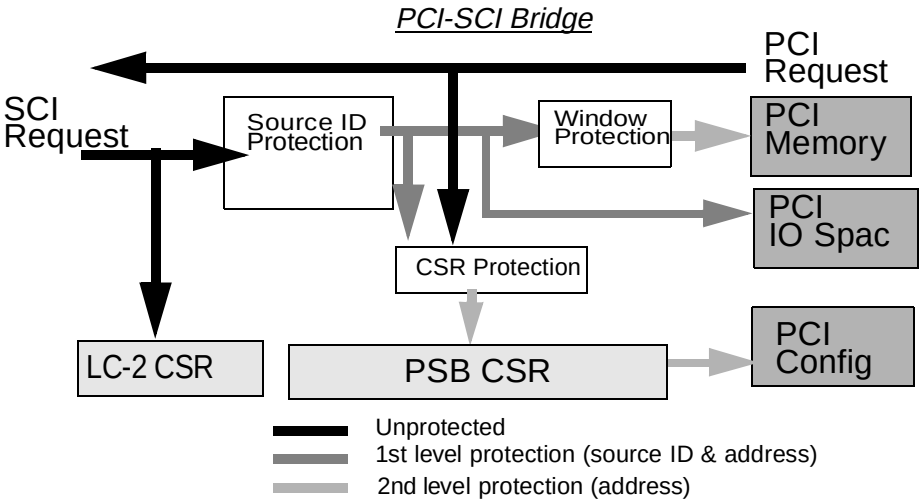


Figure 10. PCI Card protection mechanism

4.2.1 Source ID protection

All incoming request packets are checked for a valid source ID. If the packet is found ok, it can proceed to the next protection level (address checking). If the request was not ok, a response will be generated with response status RESP_ADDRESS (only response expected commands).

The PSB will use the source nodeId[11:4] as index into a 256 entry look-up table. Protection can therefore be set up for 256 groups of 16 nodes since nodeID[3:0] and nodeID[15:12] are not checked.

The source ID protection also works for SW packets.

Source ID protection override:

The source ID protection can be overridden if the incoming request's address hits the address window set up by AP_START and AP_END registers. This window has a 4 KByte granularity.

4.2.2 PCI Memory Protection

If the request packet passed the source ID check, an address validation will be done (only in 4GB range). The WIN_PROTECT register sets up two address windows of 16MB granularity; one for Read/Write permission (RW) and one for Read Only (RO) permission. Any violation will result in a response with response status RESP_ADDRESS. Address offset [47:32] is not considered in the range checking. The total access protection window must be continuous. See Figure 11..

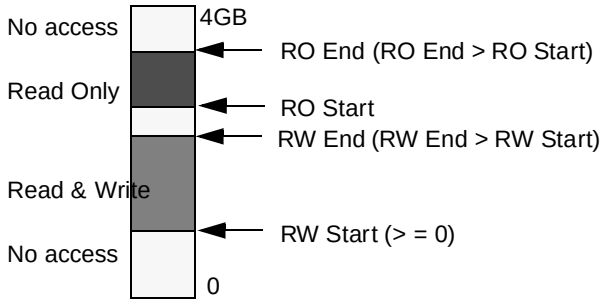


Figure 11. Window Protection setup in PCI memory space

Note: This address protection mechanism only works for PCI memory space. PCI IO space is *unprotected*. Addresses in SW packets is not checked for access violation.

4.2.3 PSB CSR Protection

This mechanism makes it possible to protect the CSR registers from write accesses from either the PCI side or B-Link/SCI side. It also provides a way to protect from nodes within the same 16-node range as the PSB64 node ID.

The CSR protection is controlled by the ACCESS_PROTECT_REG

5 CONTROL AND STATUS REGISTERS (CSR)

The CSR registers for the PCI Card are located in two different ASICs - PSB64 and LC-2. The ways of accessing them is a bit different. All the registers below can be accessed both from PCI and SCI. Access restrictions can be set up for the PSB registers. Figure 11 shows the card local mapping to the registers.

5.1 PSB64 CSR REGISTERS OVERVIEW

The layout of the PSB CSR register space defined by the CSR standard, IEEE Std 1212-1991(2), is illustrated in Figure 12.. In the following, only the register offset from the register space is used. PCI64 has a very similar register set.

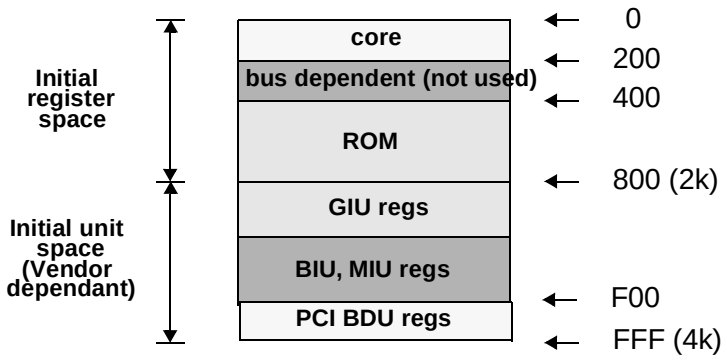


Figure 12. The CSR register space

Some core registers are mandatory, and so is the first entry of the ROM.

NOTE: The registers described in this chapter has big endian byte ordering. Accesses from the PCI bus must swap the bytes to match the little endian byte ordering on PCI. The address offset in the table below refers to start of the PSB local CSR address window (which can be accessed in different windows).

Register	Offset	Description
STATE_CLEAR	0x000	The STATE_CLEAR register provides information about the status of the PSB
STATE_SET	0x004	The STATE_SET register is used for setting bits within the STATE_CLEAR register.
NODE_IDS	0x008	The NODE_IDS register holds the current node ID and the initial node ID. The node ID is compared against the target ID of incoming SCI packets and will be assigned by software during the system initialization process.
RESET_START	0x00C	Writing the RESET_START register will initiate a immediate reset of the PSB chip

Register	Offset	Description
INDIRECT_ADDRESS	0x010	The INDIRECT_ADDRESS register will in conjunction with the INDIRECT_DATAregister form a read and write mechanism to the CSR ROM address range implemented in a NWM memory external to the PSB
INDIRECT_DATA	0x014	The INDIRECT_DATA register will perform a 4 byte read or write transaction to the external CSR ROM on a 4 byte alignment. Reading this register returns the CSR ROM[INDIRECT_ADDRESS] value in the external memory (NWM). Writing it stores the argument in CSR ROM[INDIRECT_ADDRESS] in the NWM.
INTERRUPT_TARGET	0x050	This register provides a mechanism for generating interrupts. Writing this register will set the RemInt bit in the INTERRUPT_STATUS register
INTERRUPT_MASK	0x054	This register serves as an interrupt mask for the INTERRUPT_TARGET register. A one to one correspondence exists between the 32 bits within the registers
CHIP_ID	0x3b0	Identifies the chip
CSR_ROM	0x400-0x7FF	Serial ROM
ILSTAT	0x800	Shows the current status of the interrupt lines.
EMASK	0x804	The register contains a mask bit corresponding to each bit in ESTAT register. Setting the mask bit to zero means disabling event reporting on the specified event, while setting the mask bit to one means enabling it.
EENABLE	0x808	After an event to be reported is discovered, the source will be automatically disabled after the report is sent. The source will be re- enabled by the interrupt handler to allow the source to issue a new event.
EFLUSH_MASK	0x810	Some devices always issue an interrupt after a DMA transfer is finished. This knowledge can be used to trigger a buffer flush if write gathering and invalidation if prefetching is in use. The register layout is the same as EMASK. Note that system errors can also enable flushing. The event reporting will be postponed until the flushing is finished.
EISTAT_LOW	0x820	Used when the reporting mechanism is reporting an error/ interrupt. Typical contents of these registers will be a node identification and other static information needed in the message.
EISTAT_HIGH	0x824	
EESTAT_LOW	0x828	
EESTAT_HIGH	0x82C	
EVENT_TARGET	0x830	This register contains the target destination Node ID when the event reporting mechanism is in use.
EVENT_IOFFSET	0x834	This register should contain the CSR address offset of the event report destination node when reporting interrupts.
EVENT_EOFFSET	0x838	This register should contain the CSR address offset of the event report destination node when reporting errors.
SEND_EMSG	0x840	Writing to this register sends off an event message (8 byte write selected yet) specified by the contents of the EISTAT_HIGH, EISTAT_LOW registers. The destination is specified by EVENT_TARGET and EVENT_IOFFSET.

Register	Offset	Description
ISTAT	0x900	This register contains status bits for a number of different PSB events. Each event may generate an interrupt.
IMASK	0x904	The register contains a mask bit corresponding to each bit in the ISTAT interrupt status register
ISET	0x908	The ISET register provides a way to trig a specific hardware interrupt in the PSB.
ISELECT	0x90C	This register serves as an interrupt type selection for the ISTAT register. Each bit corresponds to the interrupt sources in ISTAT. A one means that a SERR# (system error) is generated and zero means that INT (normal interrupt) is generated.
SW_FLUSH	0x910	Writing to this register will flush write buffers containing valid data. When the flushing is finished the buffers will be freed. Read value shows status of he previous flush.
SW_INVALIDATE	0x914	Writing to this register will invalidate read buffers containing valid data. Invalidation means that the read buffers are freed.
STORE_BARRIER	0x918	Writing to this register will set a store barrier for all 16 write streams. Reading this register will show which streams still have an outstanding request(s) since the last store barrier was set.
GIU_CONTROL	0x91C	Misc. control bits for PSB.
CONFIG_ADDRESS	0x920	PCI configuration cycle address
CONFIG_DATA	0x928	PCI configuration cycle data
SW_Q_START	0x930	Points to start of SW packet queue
SW_Q_END	0x934	Points to end of SW packet queue
SW_Q_HEAD	0x938	Points to head of SW packet queue
SW_Q_TAIL	0x93C	Points to tail of SW packet queue
PKT_POINTER	0x940	Points to head of SW packet to be sent.
WIN_PROTECT	0x944	Protects PCI memory
GIU_STATUS	0x94C	Misc. status bit for PSB
STREAM_ATTR	0x950	Enables use of moves
PKTRIG	0x954	Send a SW packet pointed to by PKT_POINTER.
L_ENABLE	0x958	Controls the stream combining.
S_ERROR	0x960	Shows streams in error status
STREAM_ADDR	0x964	Selects stream entry to read/write
STREAM_DATA	0x968	Contents of stream on reads and write.
ATT_ADDRESS	0x96C	
ATT_DATALO	0x970	
ATT_DATAHI	0x974	
ATC_INVALID	0x978	
ATT_DMA64	0x97C	

Register	Offset	Description
ATT_PIO64	0x980	
IO_CTRL	0x990	
IO_DATA	0x994	
SCRATCH0	0x9B0	
SCRATCH1	0X9B4	
SCRATTCH2	0X9B8	
SCRARCH3	0X9BC	
CHIP_RESET	0xD00	Different reset signals
ACCESS_PROTECT_REG	0xD04	Csr Access Protection
AP_INDEX	0xD80	Access protect
AP_ENTRY	0xD84	Access protect
BIU_CONTROL	0xD88	Misc. Control
AP_START	0xD8C	Access protection
AP_END	0xD90	Access protection
MASTER_ENA	0xF00	Enables the PCI Master after reset.
DMA_CMD	0xF04	DMA Engine command
DMA_STATUS	0xF08	DMA Engine status
DMA_LPTR	0xF0C	DMA Engine Link Pointer
DMA_SCIADDR	0xF10	DMA Engine SCI Address
DMA_PCIADDR	0xF14	DMA Engine PCI Address
DMA_CTRL	0xF18	DMA Engine Control
DMA_64BASE	0xF1C	DMA Engine 64 bit base
PCI_MISC	0xF20	
SL_DEADLK_CNT	0x7000	

Table 1. PSB's Internal Control and Status Registers

5.2 LC-2 CSR REGISTERS OVERVIEW

Table below lists all CSR (Control Status Registers) present in the LC-2 chip with respective address offset. These registers can be accessed two ways from the PCI Card. Either directly using the CSR memory window on the card or using the address mapping table in memory I/O window

Register	Offset	Description
STATE_CLEAR	0x0000	The STATE register contains the current running and error state of the node. This register address allows the clearing of certain STATE register bits.
STATE_SET	0x0004	This register address allows the setting of certainSTATE register bits
NODE_IDS	0x0008	Contains nodeId value and initial nodeId value from reset initialization.
RESET_START	0x000c	A write to RESET_START register performs an immediate command reset of the chip.
ERROR_COUNT	0x0180	Contains current count of detected errors.
SYNC_INTERVAL	0x0200	Interval between sync packets in fractions of a second.
SAVE_ID	0x0208	Unique Id bit 79:64.
ROUT_LCTBL[0:15]	0x0280-0x02bc	SCI linc-side routing table locations (16 registers, each 16-bits wide)
ROUT_BLTBL[0:15]	0x0300-0x033c	B-Link side routing table locations (16 registers, each, 16-bits wide)
ROUT_CTRL	0x0380	Configuration and control for packet routing
ROUT_MASK	0x0384	NodeID mask bits for interval routing
CONFIG1	0x0388	Contains configuration data
CONFIG2	0x038c	Contains configuration data
CONFIG3	0x0390	Contains configuration data
UID1	0x0394	Unique identifier bit 15:0
UID2	0x0398	Unique identifier bir 31:16
ELOG0	0x03a0	Register contaning a collection of error log flags.
DIA	0x03a4	Diagnostics register which provides access to performance counters., error logs, and clock strobe registers
INITST	0x03a8	Extended Hardware Init state.
VID	0x03b0	Vendor Id.(Dolphin chip Id addres)
PC_CMD	0x03b4	Performance counter command.
PC_EXT	0x03b8	Performance counter extended field.
PC_SEL	0x03bc	Performance counter selector.
PC_CNT	0x03c0	Performance event counter(clear only).
SWINFO	0x03c4	General software communication register.
SWST_CLEAR	0x03c8	Software state bitwise clear register.

Table 2. LC-2 CSR Support Registers

Register	Offset	Description
SWST_SET	0x03cc	Software state bitwise set.
BLOCK	0c03d0	Special lock register.
VIDR	0x0400	Vendor ID mirror of VID (SCI cpecific address)

Table 2. LC-2 CSR Support Registers

5.3 BOARD_CTRL CSR REGISTERS OVERVIEW

Table below lists all CSR (Control Status Registers) present in the PLD's with respective address offset. These registers can be accessed from the PCI Card using the PSB's CSR registers IO_CTRL and IO_DATA.

Register	Offset	Description
Led_CTRL	NA	Controls leds and modus
Status_B	NA	Firmware version to PLDB in the BOARD_CTRL + status
Reset _C	NA	Reset of LC2, B-link and LC2 on Daughter Card (LC2_2)
Pump_V	NA	Charge Pump Value
Enable_LC2_2	NA	Enables LC2 Daughter Card
Status_A	NA	Basic Status bits from PLDAin the BOARD_CTRL
Status_LC2_2	NA	LC2 Daughter Card status
Version_A	NA	Firmware version for PLDA

Table 3. BOARD_CTRL CSR Support Registers

.

6 PCI CONFIGURATION SPACE

6.1 OVERVIEW

The 256 byte PCI configuration space must be implemented by all PCI compliant devices and is read and written to by the Configuration Read and Configuration Write commands from PCI.

Accesses from B-Link can also access the PSB PCI configuration space. Configuration Cycles on PCI can be generated by using CONFIG_ADDRESS and CONFIG_DATA registers. The address line generating the PIDSEL (IDSEL) to the PSB must be defined but using address line 11 is recommended. This requires two CSR accesses.

The PCI specification defines that reserved bits and fields read as zero. All registers are accessible using byte, word and dword access. Registers are always accessed using one access.

Registers in the 256 byte configuration space not found below are reserved (read as zero). The layout of the configuration space isfound in Table 4. The registers are described in greater detail in the PSB64 Specification.

All PCI configuration space registers has the prefix PCI in its mnemonic to avoid confusion with the PSB64/LC2/PLD CSR registers. The CSR registers do not have any special prefix..

31	16	15	0	
Device ID(PCIDID)		Vendor ID (PCIVID)		00h
Status (PCISTS)		Command (PCICMD)		04h
Class Code (PCICLDCD)			Revision ID (PCIRID)	08h
BIST (PCIBIST)	Header Type (PCIHDR)	Latency Timer (PCILAT)	Cache Line Size (PCICALN)	0Ch
Base address register. CSR accesses. (PCIBADR0)				10h
Base address register. SCI memory. (PCIBADR1)				14h
Base address register. SCI memory , 64 bit addressing . Low (PCIBADR2)				18h
Base address register. SCI memory, 64 bit addressing . High (PCIBADR3)				1Ch
Reserved = 0's				20h
Reserved = 0's				24h
Reserved = 0's				28h
Reserved = 0's				2Ch
Reserved = 0's				30h
Reserved = 0's				34h
Reserved = 0's				38h
MAX_LAT (PCIMAXLAT)	MIN_LA (PCIMINLAT)	Interrupt Pin (PCIINTPIN)	Interrupt Line (PCIINTLIN)	3Ch
GbusConfig (PCIGBUSCONF)		Retry Timer (PCIRTRY)	Memory Size (PCIMEMSIZE)	40h
Cold Reset (PCIRST)				44h

Table 4. PCI Configuration Space Header Format

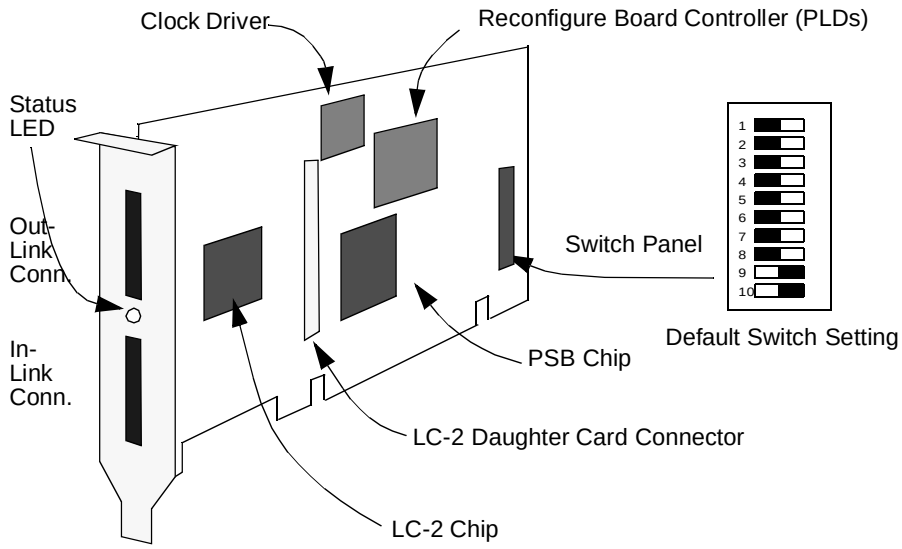
See the PSB64 Specification Document for more detailed register description.

7 PHYSICAL SPECIFICATIONS

7.1 BASE BOARD

7.1.1 Physical Appearance

The board is fullsize PCI card using 5V power and 3.3V signalling on PCI. The SCI links uses LVDS on 18 bit links. The card has one input and one output link.



The Specification on the switch panel in the figure above is given below.

1. JIGG. When on, the card may be programmed andtested via JTAG.
2. MARG LINK. When off the SCI link frequency = 100MHz. When on 125MHz
3. MARG B-link. When off the B-link frequency = 50-66MHz. When on 66-80MHz
4. FREQ B-LINK. When off, the B-link frequency = 66MHz, when on 50MHz
5. EXT RESET. When off no connection to thePCI SPARE3 pin. When on a Board Controller generated reset signal is connectedto the SPARE3 pin.
6. WATCHSEL. For testing only
7. WATCHSEL. For testing only
8. Memory size selector bit 0

9. Memory size selector bit 1. The memory size selectors determine how much PCI-memory the adapter card allocates at boot time. The table below shows the settings.

Memsize 0	Memsize 1	Memory size allocated
OFF	OFF	16 MB
ON	OFF	64 MB
OFF	ON	256 MB
ON	ON	2 GB

Table 5.

10. PCI A64. When off the card expects to sit in a 64 bit PCI bus. When on, the card expects to sit in a 32 bit PCI bus.

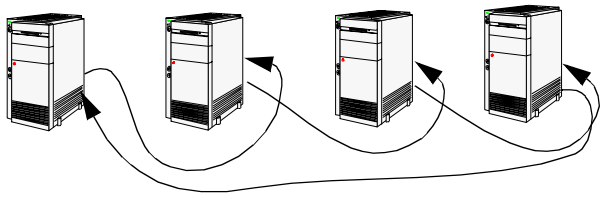
7.1.2 Topolgies

With a single-link PCI-SCI card you can connect systems in a ring. This topology is simple and requires little cabliing, but has the disadvantage that it has a single point of failure. If any of nodes goes down (power down, cable out or other errors) the whole system will go down since the SCI link is not maintained anymore.

Using a SCI switch will overcome this problem since the swith will continue routing packets even if one of several ports are down. Dolphin offers two SCI switches, one 4-ports and one expandable to 16-ports.

The simple Mask Routing is used to configure the nodes below.

Mask Routing is used in both cases.



Ring Topology

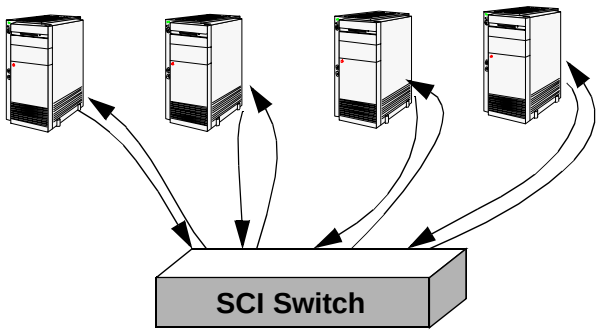


Figure 13. Fault tolerant SCI Switch Topology

7.2 OPTIONAL EXTRA SCI LINK

The PCI-64 Card also has an optional additional SCI link interface requiring one extstra slot position. This extra slot position can be a PCI or an ISA/EISA slot since the connector itself is not used. Having two SCI link on one card makes it possible to build fault toalerant system without the need of a SCI switch.

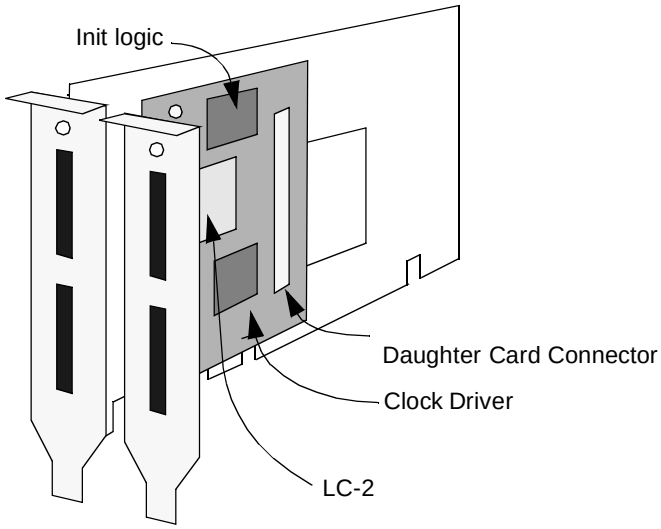


Figure 14. Dual or Triple SCI links

7.3 POSSIBLE TOPOLOGIES WITH DUAL SCI LINKS

With an extra SCI llink on the PCI SCI adapter a new set of possibilties appear. Fault tolerant topoligies can now be built without using a SCI switch.

7.3.1 Dual rings

Here the two links are just connected in parallell. This is the simplest way to connect the dual-link PCI-SCI cards and requires little SW management. If one ring goes down the other ring will used. Both fault tolerance and bandwidth increase is achieved, but it cannot withstand a dead node.

Mask Routing orTable Routing can be used together with Even/Edd Switching (transaction ID) to implement this system.

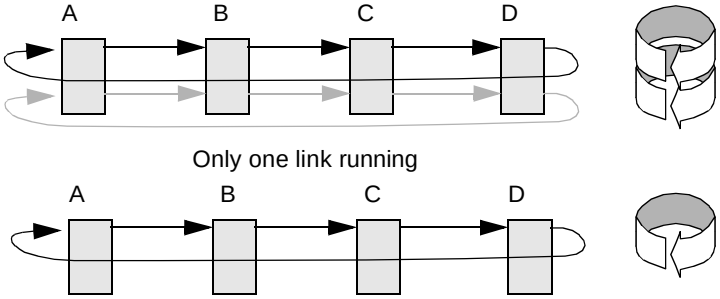


Figure 15. Dual Rings

7.3.2 Counter rotating rings

This topology also connects the nodes in dual rings. If one ring goes down the other ring will used. The idea of the counter rotating rings is also to load-balance the data transfer and optimize the use of the rings such that the link with the shortest number of bypassing nodes to the destination node is used. This system can also only toelerate one missing link.

This routing scheme requires Table Routing.

..

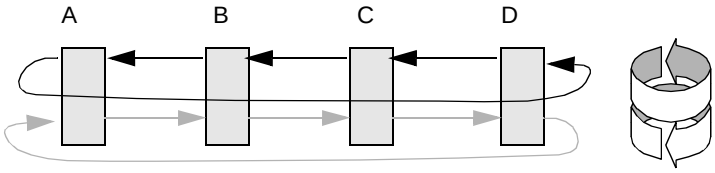


Figure 16. Counter Roating Rings

7.3.3 2D Mesh

The 2D mesh is also a powerful fault tolorant topology that can connect many nodes without the need of a SCI switch. The 2D mesh offers two alternative routes to the destination node. In the case one link goes down you can re-route the packets to the destination. This topology is much more complex to manage when it comes to error handling and reconfiguration issues.

This routing scheme requires Table Routing and the mesh is therefore limited to 256 nodes in the mesh (e.g an 8x8 mesh). An X->Y routing scheme can be used. In the mesh an alternative path can be found if one (and only one) node is dead.

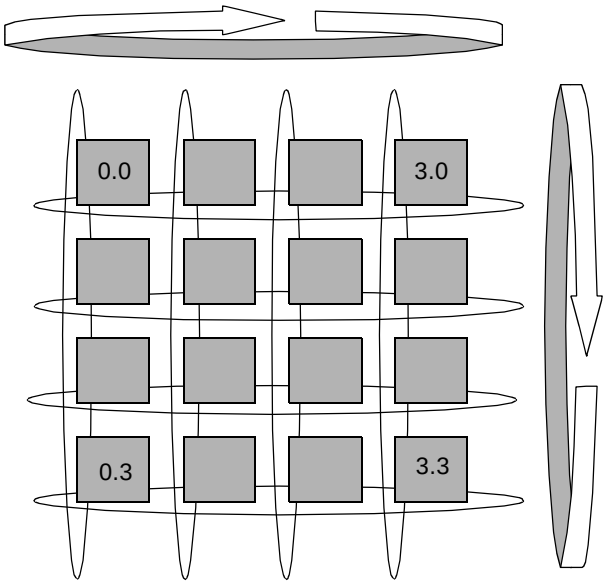


Figure 17. 4x4 2D Mesh

7.4 BOARD LAYOUT

The PCI-SCI plug-in card is a PCI Long Card.

7.5 CONNECTOR & CABLES

- 80-pin AMP connector
- Shielded twisted pair copper cables. Up to 10 m point to point.
- External fiber optic converters can be used to increase distance to 100 m and beyond.
- Cards can be connected in point to point, ring or switch topologies.

7.6 ELECTRICAL

It uses 5V power but operates both in 5Vand 3.3V signalling environments.