Tigon/PCI Ethernet Controller

Revision 1.04

ALTEON
NETWORKS

Alteon Networks, Inc.
6351 San Ignacio Avenue
San Jose, CA 95119
(408) 360-5500

Part Number 020016

August 1997

Preface

Audience

This document is intended for use by those considering and or implementing a Ethernet adapter card which is based on the Tigon PCI/Ethernet Controller. It is assumed that the reader has a basic understanding of the necessary issues related to the design of an adapter card, the PCI bus, as well as Ethernet signaling.

The following documentation should also be referenced:

Documents
PCI Local Bus Specification, Revision 2.1 PCI Special Interest Group P.O. Box 14070 Portland, Oregon 97214 (800) 433-5177
GigaBlaze™ G10™ Documentation Set, May 1997 LSI Logic Document DB14-000073-00

1 Introduction

1.1 PCI Local Bus

The PCI Local Bus is a high performance, synchronous bus, which is becoming widely accepted among system developers. The bus is capable of a maximum theoretical bandwidth of 528M Bytes per second when operating as a 64-bit bus at its maximum 66 MHz frequency. Provisions have been made to control latency at system configuration, thus allowing each machine to adjust to the flavor of adapters which have been installed.

The PCI Local Bus has also been optimized to allow direct interconnect between the bus connector and chips designed by vendors (i.e. no glue logic). The multiplexed address and data buses reduced the number of signals and pins associated with the PCI bus.

More information on the PCI Local Bus can be obtained by reading the “PCI Local Bus Specification” provided by the PCI Special Interest Group.

1.2 Performance

Performance is essential for high speed networking adapters. Many aspects of the entire system need to be considered during the design process of such an adapter. The Tigon PCI/Ethernet Controller described by this document, was designed with the following key aspects in mind:

- Intelligence on adapter to offload host
- Reduce number of interrupts
- Reduce host accesses across PCI bus
- Reduce memory to memory host copies
- Low latency
- Checksum assist
- Hardware DMA channels

The combination of all of the above aspects has lead to the architecture of the Tigon chip, as well as the host driver which interacts with both the adapter and higher layers of software within the host.

1.3 Flexibility

The Tigon PCI/Ethernet Controller incorporates a philosophy of flexibility throughout the design. It was acknowledged that as hosts develop, and their software provides additional performance oriented features, the interaction with the adapter may need to change to take advantage of such features. This philosophy lead to a design in which the primary interaction between the host on the adapter is between two processors. The host being represented as one processor and the adapter containing another. This allows to generate software upgrades which enable new or unique features between the various host platforms.

The software which runs on the Tigon's internal processor is referred to as firmware so as not to confuse it with the host driver. This firmware which interacts with the driver is actually contained within the driver and downloaded to the adapter during the initialization process. This prevents any issue of incompatible or outdated revisions between the host driver and the firmware. This also adds to the ease of installation of new software. No special firmware loading utility or PROM swapping is required.

Since the PCI Local bus is a relatively simple handshake, it is possible to create PCI to other bus bridges. These bridges can be used in conjunction with the Tigon to provide connectivity to other standard buses, or to proprietary higher bandwidth internal buses of a host.

Finally, even the Tigon itself was built around the concept of building blocks. The various interfaces provided by the chip as well as the internal processor are virtually independent objects which can be replaced or upgraded in future versions of the chip. This allows the possibility, for example, of a new chip which keeps the same host interaction, but uses a different bus interface. The internal processor could be expanded to become more powerful. Even the Ethernet interfaces can be replaced with other network interfaces.

2 Architecture

2.1 Overview

The Tigon PCI/Gigabit Ethernet Controller is designed to be a high integration and high performance Gigabit Ethernet ASIC used in adapters for workstations and personal computers. This ASIC is compliant with the PCI Local Bus Specification Revision 2.1. The Tigon contains the DMA functions necessary to off load either the host or the adapter from having to move data between the different memory spaces. A block diagram of the Tigon chip is shown on the next page.

As data is transferred between the two interfaces it is buffered in a FIFO. This allows for buffering and proper synchronization of the data to the output interface. Internally there are two FIFOs, one dedicated to each direction.

The design fully supports automatic handling of misaligned buffer transfers. This means that the data to be moved can be transferred between any two buffers regardless of the alignment of either buffer. For implementations which support the ability to DMA the data directly into the application's memory space, this feature might prevent the host from having to make unnecessary data copies. Internally, data is aligned to the FIFO boundaries.

A TCP/IP style checksum is calculated on all data that is transferred through the Tigon FIFOs. Any implementation in which the host supports hardware checksum assist can benefit from this hardware calculation. The checksum is calculated and automatically stored for each DMA buffer descriptor.

The PCI specification is inherently little-endian. Not all hosts want the data presented in the little-endian format. The Tigon contains the ability to do little-endian to big-endian swaps on either 8 byte or 4 byte boundaries. It is also possible that the Tigon can be used as a basis for non-PCI interfaces which expect the data to be in a big-endian format. This feature allows for the maximum flexibility.

The Tigon also contains a processor to manage the reading and chaining between buffer descriptors. Buffers can be chained to support the "scatter/gather" model of host memory. The code for this processor can be modified in order to optimize the format of the buffer descriptors to that which is most efficient for the host.

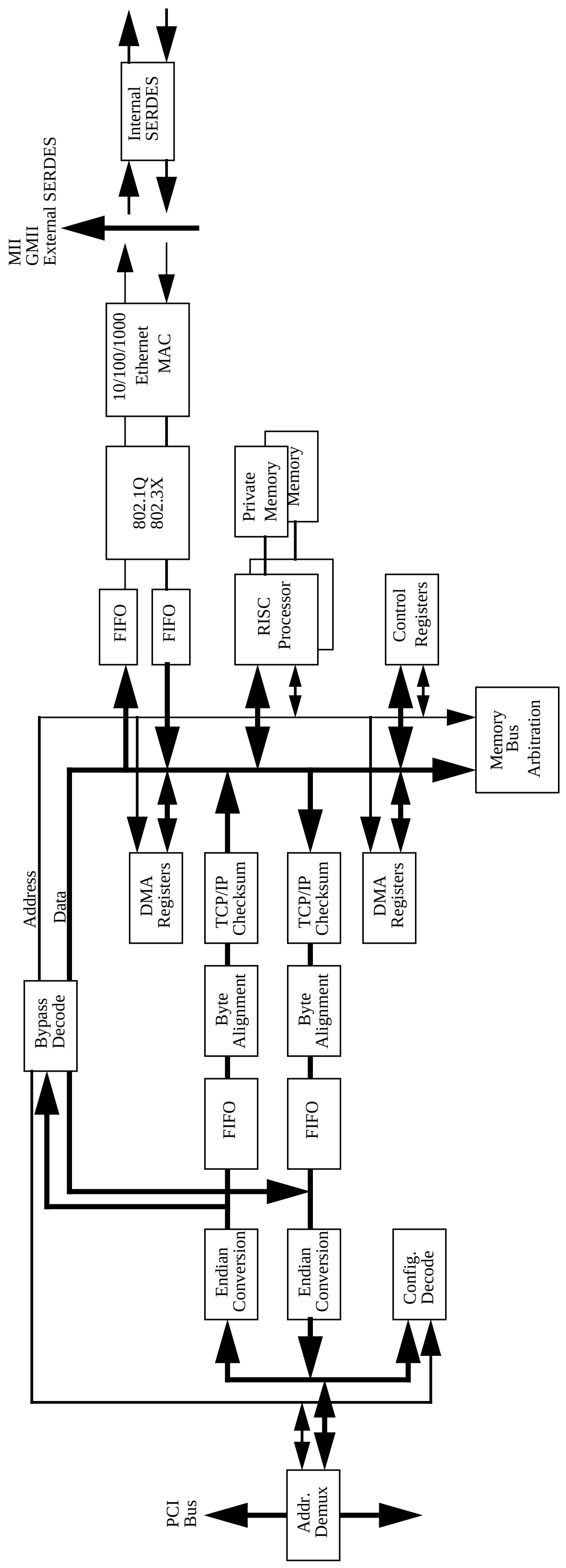

Figure 1. Tigon PCI/Gigabit Ethernet Controller Block Diagram

2.2 PCI Supported Features

2.2.1 PCI Bus Configuration

The PCI specification provides a mechanism for adapters to be uniquely configured depending on the resources available in the host in which they are inserted. In the personal computer market this feature is called "Plug and Play." The Tigon fully supports this configuration process.

At the heart of the configuration process is a maximum 256 byte region in which the adapter's PCI interface can be controlled. The adapter uniquely identifies itself and indicates to the host what it requires in terms of memory space, I/O space, bus latency, interrupts, etc. The host in return provides the adapter with the necessary information. The can host also build a data structure in its own memory which software can use to determine what adapters are in the system.

In order to simplify the Tigon PCI/Gigabit Ethernet Controller, only configuration registers are controlled by the PCI clock while all other registers are controlled by the Memory Bus clock. This enables all non-configuration registers to be accessed even if the PCI clock is not present. The actual description of these registers for the Tigon PCI/Gigabit Ethernet Controller are contained in the “Register Descriptions” chapter.

All supported configuration registers are preloaded after a reset from the serial EEPROM. This preserves the adapter configuration information even if the adapter it is moved between machines. When necessary, software can update the contents of the EEPROM if the configuration needs to be altered.

2.2.2 PCI Bus Arbitration

Since the Tigon is capable of being a PCI bus master, it participates in the bus arbitration process. The mechanism for this arbitration is outlined in the PCI specification and will not be repeated here.

The ability to arbitrate for the PCI bus and become a bus master is not enabled until the adapter is fully configured. This, however, does not prevent slave operations from accessing the Tigon. Once bus mastering is enabled it is important to note that the Tigon has no control over the arbitration algorithm used by the host. The PCI specification indicates that this is host specific.

2.2.3 PCI Bus Transactions

2.2.3.1 Read Operations

The following read operations are supported by the Tigon PCI/Gigabit Ethernet Controller:

Table 1. PCI Read Operations

Command	Description	as Target	as Master
Configuration Read	Read to Configuration region	Yes	Yes
Memory Read	Read of 2 or less words	Yes	Yes
Memory Read Line	Read of 3-12 words; or > 1/2 cache line if cache size enabled	Yes	Yes
Memory Read Multiple	Read of more than 12 words; or > 1 cache line if cache size enabled	Yes	Yes

2.2.3.2 Write Operations

The following write operations are supported by the Tigon PCI/Gigabit Ethernet Controller:

Table 2. PCI Write Operations

Command	Description	as Target	as Master
Configuration Write	Write to Configuration region	Yes	Yes
Memory Write	Write to Memory region	Yes	Yes
Memory Write Invalidate	Write a cache line to Memory region	Yes	Yes

2.2.3.3 Termination Operations

An operation on the PCI Bus may be terminated for any of several reasons. The Tigon PCI/ Gigabit Ethernet Controller responds to all of the following termination methods described in the PCI specification.

Table 3. PCI Termination Operations

Termination Method	Description
Normal Completion	Master concluded its intended transaction
Master Time-out	Grant was removed and latency timer expired
Master Abort	No target responded
Target Retry	Unable to process transaction now
Target Disconnect	Unable to respond within latency guidelines
Target Abort	Target will never be able to respond

As a target, the Tigon will never generate any of the three possible termination methods. All target operation must be terminated by the master.

2.2.4 PCI Bus Interrupt

The PCI specification allows for up to four interrupts to be used by an adapter. If an adapter uses more than one, then it must have a separate function for each interrupt. Since the Tigon PCI/ Gigabit Ethernet Controller is a single-function device it only supports one interrupt. This single interrupt should be connected to the INTA# signal on the PCI Bus.

! **NOTE:** *Some computers require interrupts to be shared with other devices, it it is left to the driver to determines if interrupts are being shared.*

All PCI interrupts are "level sensitive," and are considered asserted when low. Since interrupts may be shared, they are required to use open drain output drivers.

2.2.5 PCI Bus Errors

There are two possible types of errors defined by the PCI specification, parity errors and system errors   The Tigon PCI/Gigabit Ethernet Controller supports both of the error reporting mechanisms.

Parity errors are checked across the data bus by the receiving agent on the PCI Bus during all transactions except during a Special Cycle command. Therefore only one PCI Bus interface can drive the parity error signal at a time. This active low signal is driven by a tri-state driver which must drive the signal high again before going tri-state. This signal is synchronous to the PCI clock.

System errors are checked across the address bus and also the data bus on a Special Cycle command. Any interface on the PCI Bus can indicate a system error. This active low signal is driven by open drain drivers and utilizes a weak pull-up on the host to become deasserted. The assertion of this signal is synchronous to the PCI clock, but the deassertion should be considered asynchronous.

2.2.6 PCI 64-bit Bus Extension

The PCI specification supports a 64-bit data path in addition to the standard 32-bit data path. A total of 39 additional signals are used to provide this functionality. This extends the maximum bus bandwidth from 132 MBytes/second to 264 MBytes/second.

It is important to note that the PCI specification allows 64-bit operations with memory commands but they are not allowed for either I/O or Configuration commands.

The Tigon provides a 32/64-bit PCI interface. A protocol is defined by the PCI specification to determine what data width the host supports. Since the Tigon internally utilizes a 64-bit datapath, it was natural to build in the support for either PCI bus width.

2.2.7 PCI 64-bit Addressing

The Tigon PCI/Gigabit Ethernet Controller supports the full 64-bit addressing only as a bus master. However, the memory region required by the Tigon must be mapped into the 32-bit memory space. This implies that hosts which use 64-bit addressing must map the Tigon's memory region into the lower 4 GBytes of memory.

If utilized, the 64-bit addresses are indicated by the Tigon on the PCI bus through the use of Dual Address Cycles (DAC). The PCI specification indicates that devices capable of driving 64-bit addresses must signal a Single Address Cycle (SAC) when the upper 32-bits are all zeros in order to communicate with devices which use 32-bit addresses. This SAC technique is used by the host to access the Tigon's memory region.

2.3 Local Memory

2.3.1 Overview

The Tigon PCI/Gigabit Ethernet Controller provides a 64-bit high-speed memory bus to connect to local memory. The internal data structure of the Tigon is built around this 64-bit bus width.

The maximum frequency of the Memory Bus is 50 MHz. This allows for a maximum bandwidth of 800 MBytes/second with synchronous memory and 400 MBytes/second with asynchronous memory. Each access into the SRAM located on the Memory Bus can be to any address. Therefore no bursting is required by this memory design. The same clock which controls the local memory is used to control the majority of the Tigon chip; it is considered asynchronous from the PCI Bus clock.

Any combination of byte, half-word, or word writes are allowed by the hardware. Reads always supply 64-bits of data from the external memory. The memory sub-system has no knowledge as to whether the useful portion of the data is a byte, word, etc.

2.3.2 Arbitration

There are six possible entities within the Tigon which arbitrate for the use of the Memory Bus. A priority scheme is enforced between these six requesters so that no requester can cause a loss or corruption of the dataflow.

Table 4 lists the possible sources which can request a memory operation. A relative priority between these sources is also noted. This priority is not necessarily absolute and is discussed after the figure.

Table 4. Local Memory Priorities

Access Source	Relative Priority
Gigabit Ethernet Transmit	1
Gigabit Ethernet Receive	2
Host Slave Access	3
DMA Assist	4
PCI Read DMA	5
PCI Write DMA	5
Internal Processors	6 (3 if waiting long)

The two Gigabit Ethernet interfaces have the highest priority within the Tigon. It is imperative that the Gigabit Ethernet interfaces never underrun or overrun. Each of the Gigabit Ethernet interfaces requires attention on average every 128ns.

Host accesses to the Tigon's target logic must occur in a timely fashion and therefore it is next in the priority scheme. Such a host access indicates that there is currently no DMA activity on the bus. Depending on the level of Gigabit Ethernet activity a host access is serviced between 1 and 5 internal clock cycles.

The two PCI DMA engines alternate priority depending on which one is actively on the PCI bus. It is expected that the sustainable host memory access bandwidth will vary greatly between different host PCI interface designs. In an environment with bi-directional Gigabit Ethernet traffic, it is expected that the two DMA engines will alternate access to the host memory.

If an asynchronous local memory is used, the default priority of the internal processors can be altered via a configuration register. Normally it is considered lowest priority as long as all the other higher priority data transfers are in progress. Since the primary job of the processor is to keep the data moving, it's job is most critical when there is data to be moved which hasn't yet started. As long as any one of either Gigabit Ethernet or the PCI DMA is not active, then there will be sufficient bandwidth for the processors. Each processor is not starved for memory bandwidth since it has an instruction cache as well as a mechanism to advance to the third priority if it's request has been waiting too long.

If a synchronous local memory is used, the two processors would alternate with each other just below the DMA priority. The default priority register for each processor is ignored. To improve local memory utilization the Gigabit Ethernet, DMA channels, and CPU instruction fetch interfaces will all perform several back-to-back operations for each arbitration cycle.

2.4 Internal Processors

The Tigon PCI/Gigabit Ethernet Controller contains one or two embedded 32-bit processors depending on the revision of the Tigon. The processors can be used for any function including parsing buffer descriptors and controling the DMA hardware registers. These processors can be set up to accommodate different buffer descriptor formats, allowing the Tigon chip to adapt to the communication method which best suits the host. This approach is consistent with the goal of providing a flexible PCI Controller solution.

Fully contained inside the Tigon, the processors are capable of generating operations to the PCI Bus as well as to local memory. This allows control information to be located in either memory space. All registers internal to the Tigon can also accessed by either the embedded processors or the by the host through the Tigon's target interface.

The operation of the processors is governed by firmware. External to the Tigon will be a small non-volatile serial EEPROM which will contain all the power-on diagnostics and PCI initialization tasks. Any additional software can be downloaded by the host driver into the local SRAM memory. Since each local memory operation fetches 8 bytes, up to two processor instructions can be loaded into the Tigon on each instruction fetch cycle.

An instruction cache as well as internal SRAM is provided to enable the processor to not use any of the valuable local memory bandwidth for most firmware execution. The instruction decoding is modeled after the R4000 RISC processor with some instructions removed and several new ones added for embedded optimization.

Tigon revision 5 is the first revision to contain a second processor with its own internal SRAM. Each processor has full access to all functions within the Tigon. It is left to firmware to determine how best to utilize both processors. The model usually is either one processor manages each direction of traffic, or the one processor manages all traffic and the second processor adds higher level functions.

2.5 Events

The model with which the embedded processor executes is an Event model rather than an Interrupt model. This means that the processor is not interrupted from the task at hand, but each task should be kept to a small thread of code. There is significant hardware assist for event processing within the Tigon to facilitate a robust execution environment.

There is a single 32-bit register in the Tigon which includes bits for each of the events that the processor must handle. The processor's main loop is to wait for events and then deal with them in priority order. Two special instructions were created to allow the processor to index into an event handing jump table based on the most significant (highest priority) bit set in the event register.

Not all events are created by the hardware. There are several bits within the event register which are completely controlled by software. This allows software to queue events to be handled within the priority encoded main loop. The events in this register are described in the Register Descriptions Chapter.

One of the features that deserves special attention is the timer function. This is a 32-bit counter which advances every 1-128 local clock cycles based on a pre-scaler function. If the timer is set up to advance every microsecond, then the timer in effect rolls over every about every "70 minutes." A queue of time-based functions is maintained by the processor with the top entry in the queue managed by hardware. The processor can schedule future events by taking the current timer value and adding the number of "microseconds" at which it would like to receive an event. An event will be indicated by the hardware if the desired event time arrives or has passed within the last "minute." The processor makes heavy use of this event queue to provide time-outs, interrupts, and any other function in which it would be poor to just create a software spin loop.

2.6 Flash

The Tigon PCI/Gigabit Ethernet Controller has been designed to interface to an external flash for storage of important configuration and manufacturing information. All PCI adapters must provide configuration information to the host after power-up. The processor in the Tigon chip initially executes from this flash and loads all internal PCI registers from predetermined locations inside the flash after a reset. This enables the adapter to respond properly throughout the configuration process.

The flash technology allows vital adapter information to be accessed and updated whenever necessary. The host or local processor may read or write these locations via the Control Register. The two processors should not attempt to write the flash at the same time.

The flash has additional memory locations which are available for storage of non-PCI information. These locations can be used to store addresses, manufacturing data, diagnostic results, etc. Software is responsible for managing these locations.

2.7 Mailboxes

A common technique to facilitate communication between host processors and adapters, are mailboxes. Typically these are locations which are written by one processor which cause an interrupt to the other processor. The value written may or may not have any significance. The number of mailboxes in each direction is usually fixed. Each processor is allowed to read the mailbox only once before it is cleared by the hardware.

The Tigon has generalized the concept of mailboxes in order to provide greater flexibility in the interaction between the host and the adapter. In order to implement mailboxes in which the value of the mailbox is not significant, the processors need only define the software interrupts such that each interrupt has a predetermined meaning.

In order to implement mailboxes in which the value of the mailbox is significant, the adapter can map some of its local memory onto the PCI bus which functions as a mailbox window. This window is accessible by both processors and is predefined to be 1k bytes in size. The second 256 bytes of this window are divided into 32 8-byte locations and have a special meaning. A write to the upper half of any of these 32 locations will cause an event to the internal processor with a notification as to which location was written. Actually, there is a separate event bit for each of these 32 locations. The lower 256 bytes or the upper 512 bytes (i.e. not mailbox area) of this window have no predetermined meaning and can be software defined.

Software should be designed so that it is clear which processor is allowed to write to any given portion of the memory. This technique can be used to create many logical mailboxes of various size without being strictly limited by the hardware. An additional benefit to this technique is that the mailbox may be read as many times as necessary without the loss of the mailbox contents.

The first mailbox has special meaning since the PCI interrupt pin is immediately deasserted when it is written. This feature allows software to define a mechanism for a host processor to clear the interrupt at the same time as passing a message to the Tigon’s processor. If a hardware clear interrupt feature is not required in the software model, then this first mailbox need not be utilized.

2.8 Host Buffer Descriptors

The main mechanism to effectively communicate between the host and the Tigon PCI/Gigabit Ethernet Controller is via buffer descriptors located in host memory. This is not the only technique as mailboxes are also supported. Host descriptors are efficient since they allow for larger amounts of data to be passed without the need for host access over the PCI bus. In most host environments each time the host processor directly accesses a location on the PCI bus, the host slows down and loses some of its processing time.

It is the responsibility of the Tigon’s internal processor to use the DMA engines at its disposal to transfer the host buffer descriptors between host memory and its own local memory space. These host buffer descriptors have no correlation to the registers used to control the DMA engines. These descriptors are a software convention for communication between the host processor and the Tigon processor and are not fixed by the hardware.

The document entitled “Gigabit Ethernet/PCI Network Interface Card Host/NIC Software Interface Definition” details how the host communicates to the Tigon.

2.9 Gigabit Ethernet Buffer Descriptors

The only mechanism to queue packets to and from the Serial Gigabit Ethernet transmit and receive interfaces, is through the use of Gigabit Ethernet Descriptors. These are fixed 8-byte structures which indicate to the Gigabit Ethernet state machines where the packets are located in the local memory.

For transmit packets, the descriptors contain the starting address and length of each packet. Special provisions are included to deal with packets up to 64k bytes. For more detailed information see the “Gigabit Ethernet Transmit Interface” chapter.

For receive packets the descriptors contain the starting address and length of the packet as well as indication of any unusual or error events detected during the reception of the packet. For more detailed information see the “Gigabit Ethernet Receive Interface” chapter.

2.10 HOST DMA

2.10.1 Two Channels

The PCI interface within the Tigon PCI/Gigabit Ethernet Controller contains two independent DMA channels. One is used exclusively for host memory reads while the other is used exclusively for host memory writes. These DMA channels provide the only means by which the Tigon can access host memory.

It is the primary responsibility of the Tigon’s internal processor to manage these DMA channels for the transfer of control information as well as network data. The maximum number of bytes the hardware can transfer in a single DMA operation is 64k bytes. Due to the typical memory paging schemes used by most host operating systems, these DMA channels are typically set up by the internal processor to DMA no more than a “page” worth of data. A new host address for the next page would be required after each DMA transfer.

2.10.2 Endian Conversion

There are two basic formats for storing data in memory. One is called Little-Endian, and the other is called Big-Endian. To facilitate communication between a host which has different endianness than the adapter, the Tigon PCI/Gigabit Ethernet Controller has included the necessary byte swapping support.

The PCI Bus is inherently little endian in nature since the configuration registers are defined to be little endian. The little endian format has the lower address contain the least significant part of the field. Since it is possible that not all hosts will adhere to the little endian format for regular data transfers, or that an adapter design may be big endian, this byte swapping feature is necessary.

All data going to or from the PCI Bus, except for the PCI configuration registers, can be swapped in groups of four or eight bytes. This byte swapping is individually selectable for each of the two DMA channels as well as for any host target accesses to the Tigon.

2.10.3 Buffer Alignment

Not all DMA transfers will be nicely aligned to 32/64-bit boundaries. In fact the data could be on any byte boundary. Traditionally many adapters have placed the burden on the host to align the buffers prior to data transfer. This is time consuming and ultimately hurts system performance. The Tigon PCI/Gigabit Ethernet Controller provides the necessary byte steering logic to compensate for misaligned buffers.

The Tigon includes the logic on the PCI side of the FIFO to compensate for the misalignment. Data from any host byte address can be aligned to any byte offset of the Tigon’s internal memory. This flexible scheme allows multiple odd length, odd boundary buffers to be directly concatenated into the proper sequence prior to reaching the Ethernet MAC. This process also aids in the generation of the TCP/IP checksums.

At the beginning of all buffer transfers, the Tigon may need to move up to 7 bytes before it can begin regular full-speed bursting. The Tigon always bursts full words at a time into or out of the FIFOs. Similarly, at the end of a buffer transfer, there may be several leftover bytes which are transferred separately to complete the DMA operation.

2.10.4 Scatter/Gather

The concept of scatter/gather is very much related to that of buffer descriptors and buffer alignment. The idea is that a portion of data to be moved is to be placed in several fragments throughout memory, or may already exist in several fragments. This required the DMA operation to either scatter or gather the data respectively during the DMA process. The ability to off load this responsibility from the host processor increases the host's efficiency and overall system performance.

The Tigon PCI/Gigabit Ethernet Controller has the necessary features required to efficiently support the scatter/gather process. First, buffer descriptors need to be built which indicate where buffers are and how much data is in each buffer. In the Tigon chip these descriptors are parsed by the internal processor which manages the DMA channels. Second, the buffers may not all be on the same alignment. Again the Tigon chip can automatically compensate for such misalignments.

2.11 Gigabit Ethernet MAC

Built into the Tigon is an Ethernet MAC capable of running at either 10/100/1000 Mbits/second. Support for 10/100 modes is achieved through an MII interface. Support for 1000 mode is currently done via external Fibre Channel components operating at Gigabit Ethernet speeds.

When utilized with an external serializer/deserializer (SERDES) chip, proper Gigabit Ethernet signaling is achieved. The Tigon provides the 62.5 MHz interface required by the SERDES. The SERDES takes care of converting this into the 1.25 GHz Gigabit Ethernet signals. This division allows the Tigon to be implemented in standard CMOS technology.

The transmit and receive Gigabit Ethernet interfaces function independent of each other when configured for full-duplex operation. The MAC is also capable of operating in half-duplex operation.

FIFOs are utilized between the internal clock domain and the 62.5 MHz interface to the SERDES in order to provide data synchronization. These FIFOs are as small as possible to conserve space inside the chip.

Hardware state machines manage the transfer of packets between the local memory and the Gigabit Ethernet interfaces. All packets are queued for transmission or reception through the use of Gigabit Ethernet Descriptors. The descriptors are directly referenced and updated by the hardware. This technique provides a mechanism for the internal processor to not have to support the hardware for each packet being transmitted or received.

3 Internal Processors

The Tigon contains an internal 32-bit general purpose processor to control and manage the various DMA activities as well as to communicate with the host. This processor is a relatively independent block within the Tigon architecture. It was modeled after the R4000 RISC processor, however the supported instructions and their decoding is unique to the Tigon. Many instructions were removed and several new ones added for imbedded optimization.

The operation of the processor is governed by firmware. External to the Tigon will be a small non-volatile memory which will contain all the power on diagnostics and PCI initialization tasks. Any additional software can be downloaded by the host driver into the local memory SRAM. Since each local memory operation fetches 8 bytes, up to two processor instructions can be loaded into the Tigon on each instruction fetch cycle. A small instruction cache as well as SRAM internal to the Tigon (a.k.a Scratch Pad) is also provided to enable the processor to reduce its usage of the valuable local memory bandwidth. Firmware is compiled into the machine language opcode by using GNU R4000 tools.

Starting with Tigon revision 5, a second processor was added to the chip. To distinguish between the two processors the original one is called CPU A while the new processor is called CPU B. Both processors are equivalent with the exception that CPU B does not contain any ROM and CPU B contains less swatch pad memory than CPU A. A reset operation always halts CPU B.

3.1 Local Memory Map

The Local Memory Map defines which address ranges are to be used to access the external SRAM, external Flash, or internal Tigon registers. The local memory was implemented within a 32-bit address space. Table 5 indicates how this space was sub-divided. The regions define the maximum size provided, not all of the space need be occupied with valid locations. For example if only 256k Bytes of SRAM are assembled on the board, then only the first 256k Bytes in the External SRAM memory would be valid.

The Flash region allows for a 1M Byte of addressable space. This assumes that the full 8 byte datapath is utilized. Since most Flash implementations will use a single 8-bit wide device, the practical limit to the size of the Flash region would then become 128k Bytes.

Table 5. Local Memory Map

Address Range	Access	Max. Region Size
0x00000000-007FFFFF	External SRAM	8M Bytes*
0x00C00000-00C03FFF	Internal SRAM “Scratch Pad”	16k Bytes†
0x40000000-400007FF	Internal Bootload ROM (Instruction Fetch only)	2k Bytes‡
0x80000000-807FFFFF	External Flash	8M Bytes*
0xC0000000-C00003FF	Tigon Registers	1k Bytes

* Tigon revision 4 has 2M byte limit.

† Tigon revision 4 has 2k bytes; Tigon revision 5 CPU A has 16k bytes and CPU B has 8k bytes.

‡ CPU A only.

3.2 Processor’s Registers File

As with all processors, there are some register locations which are used only by the processor itself. These are not to be confused with the other Tigon registers which can be accessed also by the host. Architecturally there are provisions for 32 locations in this register file. Table 6 indicates the usage of these locations.

Table 6. Processor’s Register File

Register	Usage	Access
r0	Always contains 00000000h	**R/O**
r1-r31	32-bit general purpose registers	**R/W**

3.3 Instruction Set

The executable instructions for the internal processor fall into one of three decoding formats. Instructions which require an offset or a 16-bit immediate field use the I-Type format. Instructions which don’t use an immediate field or need to reference 3 registers use the R-Type format. Certain long jump instructions use the J-Type format. These three formats are shown in Figure 2.

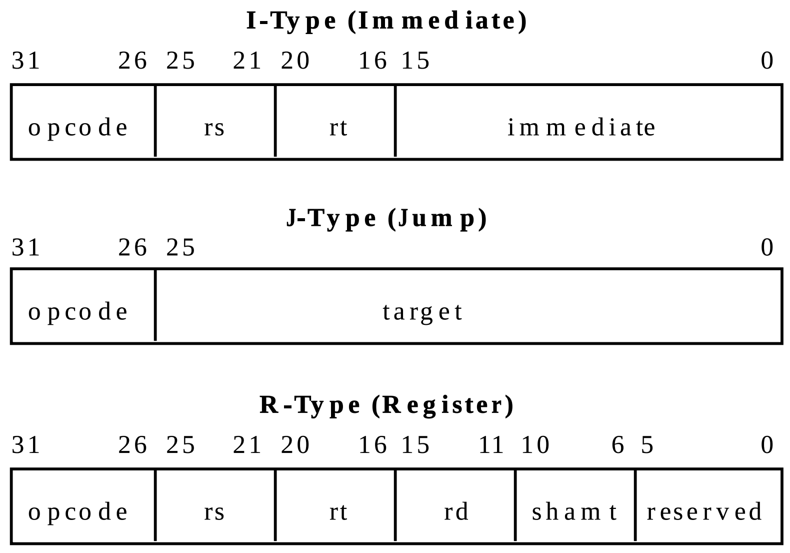

Figure 2. Instruction Formats

Table 7 lists the basic instructions supported by the internal processor. A brief description of each instruction is provided. All opcodes which are not listed will cause the processor to immediately halt.

All instructions which are listed as *new* in the following table are R-Type instructions.

Table 7. Internal Processor Instruction Set

	New	Format	Opcode [31:26]	REGIMM [20:16]	SPECIAL [5:0]	Description
Load/Store						
LB		I-Type	100000	-	-	Load byte
LBU		I-Type	100100	-	-	Load byte unsigned
LH		I-Type	100001	-	-	Load halfword
LHU		I-Type	100101	-	-	Load halfword unsigned
LW		I-Type	100011	-	-	Load word
LWL*		I-Type	100010	-	-	Load word Left
LWR*		I-Type	100110	-	-	Load word Right
SB		I-Type	101000	-	-	Store byte
SH		I-Type	101001	-	-	Store halfword
SW		I-Type	101011	-	-	Store word
SWL*		I-Type	101010	-	-	Store word Left
SWR*		I-Type	101110	-	-	Store word right
Arithmetic Immediate						
ADDI		I-Type	001000	-	-	Add immediate
ADDIU		I-Type	001001	-	-	Add immediate unsigned
ANDI		I-Type	001100	-	-	AND immediate
ORI		I-Type	001101	-	-	OR immediate
XORI		I-Type	001110	-	-	XOR immediate
SLTI		I-Type	001010	-	-	Set less than immediate
SLTIU		I-Type	001011	-	-	Set less than immediate unsigned
LUI		I-Type	001111	-	-	Load upper immediate
Arithmetic						
ADD		R-Type	000000	-	100000	Add
ADDU		R-Type	000000	-	100001	Add unsigned
SUB		R-Type	000000	-	100010	Subtract
SUBU		R-Type	000000	-	100011	Subtract unsigned
AND		R-Type	000000	-	100100	AND
OR		R-Type	000000	-	100101	OR
NOR		R-Type	000000	-	100111	NOR
XOR		R-Type	000000	-	100110	XOR
SLT		R-Type	000000	-	101010	Set less than
SLTU		R-Type	000000	-	101011	Set less than unsigned
PRI	√	R-Type	000000	-	001011	rd <- Priority Encode (rs & rt)
Shift Instructions						
SLL		R-Type	000000	-	000000	Shift left logical
SRL		R-Type	000000	-	000010	Shift right logical

Table 7. Internal Processor Instruction Set (continued)

	New	Format	Opcode [31:26]	REGIMM [20:16]	SPECIAL [5:0]	Description
SRA		R-Type	000000	-	000011	Shift right arithmetic
SLLV		R-Type	000000	-	000100	Shift left logical variable
SRLV		R-Type	000000	-	000110	Shift right logical variable
SRAV		R-Type	000000	-	000111	Shift right arithmetic variable
Jump and Branch Instructions						
J		J-Type	000010	-	-	Jump
JAL		J-Type	000011	-	-	Jump and link
JR		R-Type	000000	-	001000	Jump register
JALR		R-Type	000000	-	001001	Jump and link register
JOFF	√	R-Type	000000	-	001010	PC <- (rs[20:3] | rt[12:0]) << 1
BEQ		I-Type	000100	-	-	Branch on equal
BNE		I-Type	000101	-	-	Branch on not equal
BLTZ		I-Type	000001	00000	-	Branch on less than zero
BLTZAL		I-Type	000001	10000	-	Branch on less than zero and link
BLEZ		I-Type	000110	-	-	Branch on less than or equal to zero
BGTZ		I-Type	000111	-	-	Branch on greater than zero
BGEZ		I-Type	000001	00001	-	Branch on greater than or equal to zero
BGEZAL		I-Type	000001	10001	-	Branch on greater than or equal to zero and link
BEQL		I-Type	010100	-	-	Branch on equal likely
BNEL		I-Type	010101	-	-	Branch on not equal likely
BLTZL		I-Type	000001	00010	-	Branch on less than zero likely
BLTZALL		I-Type	000001	10010	-	Branch on less than zero and link likely
BLEZL		I-Type	010110	-	-	Branch on less than or equal to zero likely
BGTZL		I-Type	010111	-	-	Branch on greater than zero likely
BGEZL		I-Type	000001	00011	-	Branch on greater than or equal to zero likely
BGEZALL		I-Type	000001	10011	-	Branch on greater than or equal to zero and link likely
Miscellaneous						
Halt / Break		R-Type	000000	-	001101	Halt execution. A halt code can be placed in bits [26:16].

* First implemented in Tigon revision 5.

3.4 PRI Instruction

The PRI instruction is most useful for creating an index into a jump table which is required by the JOFF instruction. The result of the PRI instruction is a value between 0 - 32 corresponding to the most significant bit number which was set after the two input parameters are ANDed together plus one. The ANDing allows for masking the input so that unwanted bit positions can be ignored.

3.5 JOFF Instruction

The JOFF instruction was created to facilitate quickly processing events through the use of a jump table. Generally the PRI instruction would be used first to create an index into a jump table. The JOFF utilizes the base address of a jump table along with an index to force the program counter into the proper place in the jump table. Each entry in the jump table is 8-bytes in length as it must consist of jump instruction followed by a jump delay slot instruction (i.e., usually a NOOP).

The jump table can be of any size, but must be placed in memory such that the base address of the jump table is aligned to a address boundary corresponding to the size of the jump table. For example, a jump table containing 16 entries occupies 128 bytes of memory and must be aligned to at least a 128 byte boundary.

The *rs* field in the instruction contains the jump tables' base in the form of a byte address. The *rt* field contains the index into the table (ex. normally 0-31). It is not expected that huge jump tables would really be used since the largest number the PRI instruction can generate is 31. Bits rs[20:3] and rt[12:0] are OR'd together prior to being shifted left 1-bit position. This final shift makes each jump table entry contain two instructions.

3.6 Instruction Cache

A small instruction cache was implemented to allow tight loops to execute without accessing external SRAM as well as to take advantage of the cache line fill feature which is inherent to many cache architectures. The instruction cache is organized as a single cache line which contains sixteen 32-bit instructions. The cache will attempt to fill the remainder of the cache line unless a branch instruction forces execution to a new location. External memory is organized 64-bits wide, therefore each instruction fetch operation to external memory will yield 2 instructions.

The internal processor is capable of sustaining one instruction per cycle as long as the instructions and any data loads have cache hits. Any time either cache has a miss, there is a minimum 3 or 4 cycle delay to fetch the desired data from the external asynchronous or synchronous SRAM respectively. The delay can be longer if there is contention for the SRAM from another source. This delay is usually experienced for instruction fetches each time the processor starts executing from a new cache line. Subsequent instruction fetches within the same cache line are usually cache hits since the cache quickly attempts to pre-load the remainder of the cache line.

3.7 Data Cache

The data cache was primarily implemented as a means to store the entire 64-bit value read from external memory during load instructions. Typically the processor will have data structures greater than 32-bits, therefore the reference to consecutive words will only create a single external memory operation. The data cache is organized as a single cache line which contains two 32-bit words.

A single write buffer was implemented so that the processor store instruction need not hold up the processor while it waits for completion. If a load or store instruction is executed while the write buffer is still full the processor will stall further instruction execution until the prior write completes. This ensures operations occur in the order executed.

Any time there is a data cache miss, there is a minimum 3 cycle delay to fetch the desired data from the external SRAM. The delay can be longer if there is contention for the SRAM from another source.

Bus snooping logic was also implemented to keep the integrity of the data cache even if another source writes to a location which is currently cached. During such an event the cache will be automatically updated with the new value. This means that any processor code which is looping waiting for an external memory location to be written will not actually be generating any external memory reads.

If the Tigon is configured for 1M Byte of SRAM, then the 1M-2M Byte address region is a repeat of the first 1M Byte of SRAM. In such a configuration the snooping logic will update the data cache during accesses to the same *repeated* memory region to keep it accurate with physical memory. In other words, even though reading address zero and 1MB will yield the same result, the snooping logic will only monitor the one region which was last used to fill the data cache line.

The registers internal to the Tigon but external to the processor are never cached. They always bypass the data cache without affecting the previous cache contents. There is a minimum 1 cycle delay when loading one of these type of register.

3.8 Multi-Processor Features

Since the Tigon revision 5 introduced a second processor, a few additional features were required to facilitate proper communication between the two processors. The details on how to enable each feature can be found in the register description chapter. This section is intended to provide an overview.

Since most firmware is driven by the main event register, a main event register for CPU B was added. All of the hardware events appear in both registers and it is assumed that firmware will mask the unwanted events. Each processor also has its own software event bits. Two software events bits from Tigon revision 4 where removed in order to add the ability for a processor to send an event to the remote processor, as well as for the local processor to be able to clear the remote processor’s event.

All registers which support the debug of a processor have been duplicated for CPU B. In addition the Timer Reference register has also been duplicated allowing each processor to be able to generate its own time-based events. Each processor has its own scratch pad which appears in the same location for each processor (i.e. does not show up in the other processor’s memory map and can only be accessed using the indirect debugging techniques).

The 32 mailboxes which the Tigon supports can be divided into two groups of 16 if both processors need access to mailboxes. The lower 16 are assigned to CPU A and the upper 16 are assigned to CPU B. This feature is enabled in the Misc. Local register. If only one processor needs mailboxes, then it can use the entire set of 32 mailboxes and it is assumed that the other processor would not reference the mailboxes out of convention.

To support usage of shared registers or memory locations, a hardware semaphore was created. If firmware needs more than one semaphore, then it is assumed firmware can create more using software techniques. The hardware semaphore utilizes two register decodes, one for each processor. For a processor to obtain the semaphore it must perform a write of any data value to it's allocated semaphore register. The write must be followed by a read to determine if it succeeded. The value zero will be returned if there was no success and the value one will be returned if the operation succeeded. To clear the semaphore another write of any data value must be done by the processor owning the semaphore.

3.9 Debugging Aids

Since access to signals within the internal processor is not possible, a two-fold strategy has developed for debugging the firmware. First of all it is expected that all software will be simulated prior to executed on the actual adapter. This allows full access to any portion of the Tigon as well as creates an environment where it is easy to test unusual conditions. Secondly *hooks* have been put into the Tigon to allow special host software to act as a debugger. This allows for the ability to do simple operations in a real-time environment. This host-based debugger need not access the adapter through the driver in the event that the driver is also being debugged.

The host has access to the internal processor's Program Counter and register file at all times. This allows the host to know from where code is being executed. Another register provides the ability for the host to stop as well as single-step the internal processor (see Section 9.5.1 "Miscellaneous Host Control"). Once the internal processor is stopped, then the host can even write to the Program Counter register in order to force execution to start at another location.

3.10 Reset

A hardware reset can be initiated by the PCI reset signal, the Misc. Host register, or the watchdog timer. All sources force the CPU to follow the same sequence of events. A reset does not alter the values of the register file or the internal scratch pad, but any subsequent ROM code will alter some of the locations in those memories.

If the revision of the Tigon contains more than one processor, than both will be reset at the same time. Only the processor with ROM code will start execution after the reset is completed.

3.11 Watchdog Timer

Each processor in the Tigon has its own watchdog timer feature to detect when the software is no longer operating normally. This feature can be enabled or disabled in the CPU State register and is disabled after a reset.

The watchdog state machine uses bit-19 of the Timer register to determine if the CPU needs to be reset. If the timer is set up to operate at the suggested 1 microsecond intervals, then bit-19 will toggle about every 0.5 second. The watchdog state machine looks for three transitions of the timer bit without having a write occurring to this Watchdog Clear register in order to reset the CPU. The 3 transitions ensures that reset would occur between 1 - 1.5 seconds after the last Watchdog Clear write. The data value written to the Watchdog Clear register is irrelevant.

If a Tigon revision which has more than one processor is being used, then it should be noted that either processor's watchdog timer expiring, will cause both processors to be reset. A watchdog reset does not affect the operation of any portion of the Tigon outside of the processors.

If software wishes to determine if the CPU was reset by the watchdog timer, then a bit not present in the CPU register region which normally would be preset/cleared by a full reset can be used. This bit can be set to the non-reset value during initialization. If prior to initialization this bit is in the non-reset state, then only the CPU must have been restarted. A more complex algorithm may be needed depending on if there are other reasons the software may have been restarted.

3.12 ROM Code

The Tigon contains several hundred instructions of on-chip code in read-only memory (ROM) that the processor[1] executes when it first comes out of reset. The purpose of this code is to locate, load, verify, and run a program from external storage, thus providing the ability for the Tigon to boot without the aid of another processor or external logic. Bootload code can be found in either serial EEPROM, or a parallel storage device mapped into the Tigon's flash memory space. Priority is given to the serial EEPROM. The flash memory space is only checked if attempts to read from serial EEPROM fail.

As the ROM code executes, the top 4 bytes of internal scratchpad is used to indicate its progress[2]. This provides a means of determining the progress of the ROM from an external host. The ROM progress code is located at address 0x000C03FFC (in the local processor's address space) and the values that it takes are defined in Table 8.

Table 8. ROM Progress Code

Progress Code	Description
1	ROM is attempting to read from serial EEPROM.
2	ROM is attempting to read from flash.
3	ROM found neither serial nor parallel device.
4	CRC of bootload structure failed.
5	ROM is loading code (from either serial or parallel device).
6	ROM finished ok, ROM_FAIL bit has been cleared.

When ROM begins to execute, it first attempts to read the magic value from serial EEPROM offset x10. If successful it reads the next four 32-bit words and checks the CRC. If the CRC passes, ROM loads code from serial EEPROM into SRAM and runs it, clearing the ROM fail bit. If any errors happen during the loading of code, the ROM halts. If either the CRC at offset x20 fails or the magic value is not found, the ROM code switches to try and load code from parallel space. Here the same procedure is repeated. If the magic value is not found or the CRC fails or an error happens while loading code, the ROM code halts.

Version 4 and earlier of the Tigon had a ROM_FAIL bit in the CPU state register that was set by hardware during reset and cleared by ROM after loading code. Version 5 of the Tigon ASIC provides the same functionality as well as the ROM progress feature for backward compatibility.

[1]Starting with Tigon revision 5, only CPU A can execute from ROM.
[2]First implemented in Tigon revision 5.

On PCI bus-based systems, it may be desired to program PCI registers like the PCI device and vendor ID registers before the PCI BIOS accesses the Tigon. In this case the process described above can be used to load a short program into the Tigon's internal SRAM. This code would perform any time-critical setup and then load and run another program containing diagnostics or more boot code. This method of staged bootstrapping can also be used to load code that configures other parts of the Tigon such as the speed that flash is accessed or the type and configuration of external SRAM memory.

3.13 Bootload Data Structure

A serial EEPROM or parallel storage device is bootload-able if it adheres to the following structure. Byte ordering is big-endian. Each field shown below is 32-bits and all other offsets are don't care. Notice the first 16 bytes were skipped in order to avoid any potential overlap with pre-defined addresses within flash memory components.

Table 9. Bootload Data Structure

Logical Byte Offset	32-bit Value
0x10	Magic value of 0x6699AA55.
0x14	Location in Tigon's address space to place code (int. or ext. SRAM).
0x18	Code length in 32-bit words.
0x1C	Logical byte offset within device where 1st byte of code is located.
0x20	32-bit CRC, calculated for addresses 0x10-0x23 (inclusive) for a total of 20 bytes. Make sure CRC is stored with big-endian byte ordering.

3.14 Accessing Serial EEPROM

Serial EEPROM, along with any I^2C compliant device, may be connected to the Tigon using two pins; a data pin and a clock pin. These pins are controlled by bits in the Misc. Local register. There is no hardware support for accessing serial devices on these pins. The Tigon ROM can access one serial EEPROM device of up to 64K bytes of memory.

3.15 Accessing Flash

The Tigon supports an 8-bit wide parallel device such as an EEPROM or flash. If present, this device is mapped into the Tigon's Flash memory space and is connected to the most significant byte (bits 63-56) of the local memory bus. Because the parallel storage device is mapped into a single byte-lane on the Tigon's 64-bit memory bus, consecutive byte addresses in the parallel device exist eight bytes apart in the Tigon's memory space. As done in the ROM, software must take this into account when accessing this space. If a wider device is used, ROM will not take advantage of the additional width, though subsequent software may choose to do so.

Access timing for the flash memory space is programmable but defaults to a very slow timing which can accommodate a device with a 180ns or faster access time. For more details on altering the speed of accesses to the flash memory region see the Fast_Flash bit in Section 9.5.2 "Miscellaneous Local Control" register and, starting in Tigon revision 5, the flash fields in Section 9.7.12 "CPU Priority" register.

Writes to the flash memory region are ignored unless the Enable_Flash_Writes bit has been set in the Misc. Local register. This is done to help prevent inadvertent writes to non-volatile memory by the host or local processor.

4 Internal Events

The Tigon PCI/Ethernet Controller contains a Main Event register which assists the internal processor in determining what task to perform next. Both hardware and software events are indicated in this event register. The processor's main loop is to wait for an event and then deal with them in a priority order. Two special instructions were created to allow the processor to priority encode the event register and use the most significant (highest priority) bit set to index into an event handing jump table. These instructions are the PRIO and JOFF instructions.

For Tigon revisions which contain two processors, there are two Main Event registers, one for each processor.There are common hardware events, but each has its own software events. There are also interprocessor events to facilitate communication between the two processors.

Intermixed with the hardware event bits are bits which are not preset by hardware. These additional bits are entirely managed by firmware as a means of indicating additional events which need processing. These bits are spread out throughout the event register as a means of allowing firmware to utilize various priority levels for these firmware events.

Event bit positions which are to be ignored can be masked off by the firmware. The priority encode instruction provides a means to specify a mask register which gets ANDed with the register which is to be priority encoded. In this manner firmware can manage a mask register.

The layout of the Main Event register can be found in Section 9.5.8, "Main Event A/B."

4.1 Timer Event

One of the events worth special attention is the timer function. This is a 32-bit counter which advances every 1-128 local clock cycles depending on the value of a pre-scaler. If the timer is set up to increment every microsecond, then this counter in effect rolls over every about every 71 minutes. A queue of time-based functions is maintained by the processor with the top entry in the queue managed by hardware. The processor can schedule future events by taking the current timer value and adding the number of "microseconds" at which it would like to receive an event. This value should be written to the timer reference register so that an event can be generated when the timer is equal to the timer reference. The processor makes heavy use of this event to provide time-outs, interrupts, and any other function in which it would be bad to create firmware spin loops.

4.2 Mailbox Events

This event indicates that a mailbox location was written to using the target interface on the PCI bus. Since there are 32 possible mailbox locations, a separate Mailbox Event register was created which feeds into the main Event register as a single bit. Any bit set in the Mailbox Event register will cause the corresponding indication in the Event register. If firmware wished to mask off certain mailbox events, then this masking function need by done by the firmware prior to using the mailbox event register. Mailbox events are cleared by setting a "1" in each bit position to be cleared as a write is performed to the Mailbox Event register.

To support multiple processors, the 32 mailboxes can be split into two groups of 16 events. Each group feeds a different Main Event register. This allows each processor to manage its own private mailbox events. This option is further defined in Section 9.5.3, “Miscellaneous Configuration.”

4.3 UART Events

The UART which is built into the Tigon generates events as a means to indicate that a byte has been transferred. The UART operates in a full-duplex capacity, so as soon as the event is generated another byte can begin transferring. There is an event for byte transmission, byte reception and the receive byte FIFO has overrun causing the loss of at least one byte. These events are defined in Section 9.5.2, “Miscellaneous Local Control” register.

4.4 PCI Interrupt Event

The PCI Interrupt Event is a hardware maskable event which allows the Tigon to respond to the assertion of the PCI Interrupt pin. This event would not be used when the Tigon is used as a normal network interface card, but rather in custom applications. If enabled, this event remains active until the input pin gets deasserted. The enable bit appears in Section 9.5.3, “Miscellaneous Configuration” register, while the event appears along with the UART events in Section 9.5.2, “Miscellaneous Local Control” register.

4.5 DMA Completed Events

There are two events associated with DMA completion. One for the read DMA channel and one for the write DMA channel. These events indicate that a host read or write DMA has completed. These events are always asserted as long as the respective DMA channel is not active. If the firmware chooses to directly control the starting of the DMA channels, then it would reference these events. Otherwise the Assist logic should be activated to control the starting of the DMA channels through the use of DMA descriptors in memory. With the Assist logic enabled, these events should be masked off by firmware.

The DMA completed events only indicates DMA operations which have completed successfully. See the respective attention events for the reporting of DMA errors.

4.6 Assist Completed Events

There are four event associated with Assist Completion. A high and low priority for each of the read DMA and write DMA channels. These events are only allowed to be generated if the Assist state machine has been enabled to control the DMA channels based on DMA descriptors in local memory. The assist logic supports 16 DMA descriptors in a circular ring fashion in local memory for each of the four events. A producer, consumer, and reference registers are used to manage each ring. At any time the reference register is pointing to a descriptor which has been completed by the DMA channel as indicated by the producer advancing past the descriptor, but not yet acknowledged by the firmware as indicated by the consumer not yet reaching the descriptor (i.e. reference not between the producer and consumer, nor equal to consumer), then the respective event is generated. The high priority rings are always serviced prior to the low priority rings.

4.7 DMA Read Attention

This event is used to indicate than the DMA stopped due to an error. This event is not present if a host read DMA is active. The exact cause of the error can be determined by reading the DMA Read State register. This event has a higher priority than the DMA Read Completed so that firmware can chose to use this event as a stopped due to an error indication. The conditions for generating this event can be found in the upper byte of the register defined in Section 9.6.3, “DMA Read State.”

4.8 DMA Write Attention

This event is used to indicate than the DMA stopped due to an error. This event is not present if a host write DMA is active. The exact cause of the error can be determined by reading the DMA Write State register. This event has a higher priority than the DMA Write Completed so that firmware can chose to use this event as a stopped due to an error indication. The conditions for generating this event can be found in the upper byte of the register defined in Section 9.6.4, “DMA Write State.”

4.9 Ethernet Transmit Event

This event is used to indicate that a particular packet or group of packets has been transmitted. This event can be disabled if it is not desired to know on a per packet basis when to free space in the transmit buffer. A common technique to avoid this event is to calculate the actual available transmit buffer space only when the firmware believes it is running low.

4.10 Ethernet Receive Events

There are two events associated with the reception of Ethernet packets. One indicates that a packet has started and the other indicates that a packet has completed. This allows for firmware to decide whether it wants to operate in a store-and-forward mode, or start processing of the packet header as soon as the header arrives. The completed event is a higher priority than the packet started event so that the firmware can be optimized for the case when the firmware is at least one packet behind in its processing and it can handle the entire packet at once.

4.11 Ethernet Transmit Attention

This event is used to indicate that there was an exceptional condition on the transmit interface that needs firmware attention. This can be either a fatal error, or a 802.3x flow-control packet was received and the firmware enabled them to generate attentions. The conditions which generate this event are described in the upper 8-bits of the register defined in Section 9.11.1, “Ethernet Transmit State.”

4.12 Ethernet Receive Attention

This event is used to indicate that there was an exceptional condition on the receive interface that needs firmware attention. This can be either a fatal error, an 802.3x flow-control packet was sent, or a low buffer/descriptor threshold was reached. The latter two conditions can be individually disabled if so desired. The conditions which generate this event are described in the upper 8-bits of the register defined in Section 9.11.7, “Ethernet Receive State.”

4.13 Software Events

The Software Events are not predefined by the Tigon design. They consist of bits within the event register which are entirely managed by firmware. The higher the bit within the event register, the higher the associated priority of the task designed to handle the event.

There are a total of 14 software events supported by the Tigon. The firmware should manage these bits as ordinary read/write bits.

4.14 Inter-Processor Events

Starting with Tigon revision 5, there are two processors in the Tigon ASIC. To facilitate the ability of one processor to signal an event to the other processor, two software events where redefined. These two events function much like software events except that if the Set Remote CPU Attention bit is written in one main event register, it appears as the Remote CPU Attention bit in the other main event register (and vice-versa). To clear the Remote CPU Attention bit, that bit position must be written with a “1.”

5 PCI Interface

The chapter defines the important issues and features of the Tigon's PCI interface. The Tigon acts as both a target as well as a master on the PCI bus. Each of these two interfaces was implemented separately in the Tigon and therefore is documented separately in this chapter.

5.1 Target

The Tigon acts like a target on the PCI bus during configuration as well as when the 16k byte shared memory region is being accessed. A common state machine handles both types of operations. Target operations are vital to the functionality of the Tigon, despite the fact that most data is transferred while in the bus master mode.

5.1.1 32-bit Address

The base register for the shared memory region is 32-bits in width. This means that the PCI configuration process will allocate an address to the 4k byte shared memory which is in the lower 4G byte of the address space. For 32-bit hosts, this process is natural anyway.

5.1.2 Synchronization

The PCI clock frequency is defined as being between 0 and 66 MHz. Since the exact value is not known, the Tigon was designed to operate the majority of its logic off of an independent clock source. This additional clock source is therefore considered asynchronous to the PCI clock.

All PCI control state machines as well as the PCI configuration registers are controlled based on the PCI clock. Any target access to other registers requires that the datapath control signals be synchronized between the two clock domains. This process is entirely handled by the Tigon. During bus master operations, the DMA FIFOs operate as a buffer between the two clock domains. This allows for full PCI transfer rates despite the synchronizing.

5.1.3 Burst Length

Target operations to the Tigon were not intended as the primary means to transfer data between the host and the local memory. As a result a limit was placed on the length of a burst supported. If the originator of the burst does not terminate the burst prior to an 8 word boundary, then the Tigon will terminate the operation at that time. This issue is transparent to software and does not affect driver operation.

5.1.4 Wait States

The Tigon will insert wait states for the operation on the PCI bus while it attempts to complete the transaction. Due to the need to synchronize the clocks as well as internal resource contention, the exact number of wait states cannot be predicted. The Tigon will not take longer than 16 clocks to respond to target requests used during normal operation. There are some diagnostic cases in which the response time can theoretically be a little longer than 16, but these operations do not reflect the normal use of the Tigon.

Simulation shows that for typical operations to registers internal to the Tigon, between 2-6 wait states are inserted. For operations to the local memory external to the Tigon between 5-10 wait states are inserted. In both cases the average number of wait states is the number in the middle of the respective ranges and is affected by the speed of the PCI bus.

Accesses from the PCI bus directly into the DMA FIFOs was not intended for use other than for diagnostics. This method uses the high speed datapath synchronizers to achieve a zero wait state access. Accesses directly to the FIFOs should only be performed when the bus master interface has been disabled.

5.1.5 Watchdog Feature

To protect against unforeseen situations in which a target operation to the Tigon causes the PCI bus to hang, a watchdog feature was built into the chip. This watchdog will automatically issue a retry in the event that the Tigon takes longer than 128 clocks to respond to a target request. This scenario can only occur due to a broken adapter since the Tigon would not normally take longer than about a dozen clocks maximum to respond to any target operation.

5.1.6 Terminating Conditions

PCI bus operations can be terminated by either the master or the target for a number of reasons. All of the mechanisms for such termination are defined in the PCI specification. This section defines when the Tigon will use each of the target termination techniques.

5.1.6.1 Target-Abort

A target-abort is issued under only one circumstance and should be considered fatal to the operation of the adapter. In all cases no data will transfer at the time the target-abort is issued. The target-abort is primarily for the benefit of the Tigon. If at any time the bus master who initiated the operation disappears from the bus in the middle of an operation, then the Tigon issues an immediate target-abort as a means of properly concluding the operation. The bus master is considered absent if neither FRAME# nor IRDY# are being asserted. This scenario would probably only occur due to faulty hardware or if a bus master has the ability to reset itself without first properly terminating all PCI operations.

5.1.6.2 Disconnect

A disconnect is issued under three circumstances. In all cases the word being transferred will complete prior to the transaction being terminated. The first circumstance is when the bottom two address bits are not ‘b00 and therefore indicate that something other than a linear burst order is desired. The majority of target operations to the Tigon are single word operations and therefore are not affected by burst order.

The second source of a disconnect indicates either a faulty bus master or an unsupported operation to the Tigon. The disconnect would be issued by the Tigon when it does not support the command operation indicated.

The third source of a disconnect indicates that a burst was attempted past an 8 word boundary. The Tigon will terminate the transaction after the current word is transferred so that the bus master may start a new burst to continue the transfer of data.

5.1.6.3 Retry

A retry is issued under two circumstances. In all cases no data will transfer at the time the retry is issued. The first circumstance is when an operation is performed directly to one of the DMA FIFOs when that FIFO is not ready for the operation. For example, this would occur if a read is performed and no data is in the FIFO, or a write is performed when the FIFO is full. Since this type of an operation is intended primarily for diagnostic purposes, it does not affect normal Tigon operations.

The second source of a retry is when the target watchdog timer expires. This function is described in more detail in Section 5.1.5, "Watchdog Feature."

5.2 Master

The Tigon is capable of being a high-speed bus master on the PCI bus. This method is intended to be the primary means in which data is transferred between the host memory and the local memory. The following categories characterize the capabilities of the Tigon as a bus master.

5.2.1 Dual Channels

There are two DMA channels built into the Tigon. One is used exclusively for reads while the other is used exclusively for writes. Each is capable of sustaining bursts at the maximum rate allowed by the PCI specification.

Separate FIFOs are used for each DMA channel as an elasticity buffer between the PCI interface and the local memory interface since each of those two interfaces operates at its own frequency. For Tigon revision 4, the DMA channels have a 256 byte FIFO while Tigon revision 5 expanded the FIFOs to 512 bytes. The FIFOs are organized as 64-bit entries to match the local memory width as well as the maximum PCI width.

5.2.2 64-bit Addresses

The Tigon supports a 64-bit host address environment. If 32-bit host addresses are desired, then the upper 32-bits of the address should be set to zero. Addresses are driven onto the PCI bus by either using a single or dual address cycles as defined by the PCI specification. At any time if the upper 32-bits of the host address are equal to zero, then a single address cycle is generated on the PCI bus.

5.2.3 Burst Length

There are many factors which determine the length of a burst on the PCI bus. Many factors are system dependent and are outside of the control of the Tigon. This section helps indicate what features have been built into the Tigon to help manage burst length for each DMA channel as well as fairness between the two DMA channels.

Generally speaking, if both DMA channels are active, they will alternate in their usage of the PCI bus. Each channel will utilize the bus for the longest burst possible until a termination condition occurs. At which point the other DMA channel will be the next in line to use the bus.

To help ensure fair usage of the PCI bus between the two DMA channels, the concept of minimum and maximum length transfers were introduced. These features are intended merely as guides which can be used to tune performance.

5.2.3.1 Minimum Burst Length

The minimum burst length indicates how many words each DMA channel will be allowed to transfer before the other DMA channel will be permitted to contend for the PCI bus. This situation can be best understood with the example of a DMA transaction being terminated just after it started. It would be *unfair* to give the next transaction to the other DMA channel.

Any number up to 255 words can be used as the minimum burst length. A value of zero disables this feature. If a DMA transaction has already terminated having transferred within 7 words of the minimum burst length, then the other DMA channel will be allowed to use the bus even though the exact minimum value has not yet been reached. This is an attempt to avoid unusual lock-step type operation in which a DMA channel gets the bus for only a couple of words before having to give up the bus anyway. The PCI State register is used to enable this feature.

5.2.3.2 Maximum Burst Length

The maximum burst length indicates at which address boundaries the DMA channel must terminate. This can either be used because the system does not want to transfer over a particular address boundary such as a page boundary, or to limit the length of a single PCI burst transaction in order to allow the other DMA channel a chance to use the bus.

This feature is really more dependent on the host address than the length of the transfer, but it does limit the maximum burst length. For long DMA operations it has the effect of setting the average maximum number of words each DMA channel can use the PCI bus before allowing the other channel to use the bus. This feature does not have much of an effect in a system environment in which the bus is frequently taken away from the Tigon to be given to another bus master.

Any of the following byte address boundaries can be used to terminate a DMA burst: 4/8 (depending on bus width), 16, 32, 64, 128, 256, or 1k bytes. This option can also be enabled or disabled in the PCI State register.

5.2.4 Wait States

As a DMA master, the Tigon never inserts wait states. This simply means that the memory system which is acting as a target will be the gating factor in determining the sustained DMA performance. During writes, the first data word is available on the clock cycle following the address. It will remain on the bus until the target accepts the data.

5.2.5 Terminating Conditions

PCI bus operations can be terminated by either the master or the target for a number of reasons. All of the mechanisms for such termination are defined in the PCI specification. This section defines when the Tigon will use each of the master termination techniques.

5.2.5.1 Completion

Completion refers to the normal termination condition in which the master has completed its intended transaction.

5.2.5.2 Time-out

Time-out refers to the situation in which the master’s GNT# line is deasserted and the latency timer has expired. The Tigon will terminate the PCI transaction after the next data transfer. This situation does not necessarily mean that the original intended transaction has been completed. The Tigon may re-request the PCI bus in order to complete any additional transactions.

5.2.5.3 Master-Abort

A master-abort is issued under the one circumstance when no target responds with a DEVSEL#. This means that the target is not responding or that the address used was invalid. The DMA channel which generated the master-abort will indicate an attention to the internal processor and will then cease all DMA activity until it has been re-enabled by the firmware. This allows for firmware to deal with this error recovery situation.

5.2.6 Read Commands

The PCI specification defines three different types memory read commands. They are used to distinguish between the amounts of memory which is to be read. This allows the host memory sub-system to prefetch words from memory prior to them being read. The Tigon also uses these commands as indications to the number of words. In no case are the commands to be used as an advance indication to the exact number of words which will be transferred.

Table 10 defines which command will be used depending on the number of words to be read. Notice that the Tigon can alter which commands are used during medium sized transfers. This is done by setting a configuration bit in the PCI State Register and was done to best be able to alter the Tigon behavior in different systems to try to achieve the highest possible performance.

Table 10. PCI Memory Read Commands

Number of Words	Normal Command Set	Optional Command Set
1-2	Memory Read	Memory Read
If cache size disabled 3-15, else > 1/2 cache line	Memory Read Line	Memory Read Multiple
If cache size disabled >16, else > 1 cache line	Memory Read Multiple	Memory Read Multiple

5.2.7 Write Commands

The PCI specification defines two different types memory write commands. They are used to distinguish between the amounts of memory which is to be written. This allows the host memory sub-system to optimize for the size of the transaction.

Table 11. PCI Memory Write Commands

Number of Words	PCI Command
If cache size disabled, or length < cache line	Memory Write
If cache size enabled, and length = multiple of cache lines	Memory Write and Invalidate

5.3 Extensions

Several extensions has been implemented to the PCI interface. These extensions were not intended to be utilized unless the Tigon was placed on a special PCI bus which needed the extensions. It was envisioned that the Tigon would be used with some additional external logic to interface directly to proprietary host internal buses. For these models full conformance with the PCI specification would not be necessary.

5.3.1 66 MHz PCI Clock

The Tigon revision 5 was designed to operate with a PCI clock up to 66 MHz.

The Tigon revision 4 was designed to operate with a PCI clock up to 50 MHz. This extension is possible only if the host PCI system can reduce its requirements for PCI timing by the difference between 33 MHz and 50 MHz. The setup and hold time required by the Tigon remains the same as the PCI specification indicates for 33 MHz operation.

5.3.2 PCI ROM

The Tigon was designed to support a fixed 32k Byte PCI ROM region. The standard PCI ROM Base address is used to map this region into the host’s address space. This region directly maps to the SRAM external to the Tigon between 0x08000 - 0x0FFFF. The PCI ROM region can be enabled at the same time the standard base address is enabled without any negative consequences.

Tigon revision 4 always requests the host BIOS to allocate the ROM region, whereas Tigon revision 5 can completely mask off the request for any PCI ROM region or force PCI retries while the local memory is being loaded. See Section 9.5.7, “PCI State” for more information on the control bits for the PCI ROM region.

5.3.3 Alternate PCI Commands

In board level products which use the Tigon with some external logic to connect to a non-PCI bus, it may be useful to have the Tigon provide a completely new set of “PCI” commands rather than those defined by the PCI specification. The Tigon can provide such an alternate set of commands which provide greater insight into the length of the DMA. All eight of the PCI command bits are used in these alternate encodings regardless of the bus width.

These alternate commands work best in environments where it is desirable to have access to the count of the number of words to be transferred. For Reads the Tigon will indicate the maximum number of words left to be read in the DMA, while for writes the Tigon will indicate the number of words available in the FIFO. It is up to the external logic to terminate the burst if it so desires prior to reaching the end of the DMA operation.

Table 12. Alternate Command Encoding

Bits	Field	Description
7	Write Operation	High for writes, low for reads.
6	Number of Words > 63	Set if there are more words than bits 5-0 can indicate.
5-0	Number of Words	Least significant 6-bits of: For writes, number of 32-bit words in FIFO. For read, number of 32-bit words left in DMA.

5.3.4 PCI Interrupt as an Input

The standard PCI interrupt signal PIntA can also be optionally used as an input which generates an internal firmware event. This feature would only be utilized in special applications in which the PIntA pin is not used as a normal interrupt to a host processor. The enable bit for this feature can be found in Section 9.5.3, “Miscellaneous Configuration.” The event which gets generated is described in Section 9.5.2, “Miscellaneous Local Control.”

5.3.5 Additional PCI Arbitration Output

To facilitate PCI arbitration when a custom PCI bridge is implemented, an additional optional output can be enabled to inform the arbiter whether the requested operation’s destination is on the local bus or on the other side of the bridge. By assigning the address space such that all devices on one side of the bridge have address bit-31 low and all devices on the other side have address bit-31 high, a single bit can be used to determine the need to cross the bridge. When enabled the Tigon would use the PIntA pin as an output which drives address bit-31 to the arbiter while the PCI request is asserted. The PIntA pin cannot be used for its normal interrupt function when this features is enabled.

To enable this feature the PCI BIST register needs to contain the value 0x7E (see Section 9.4.10, “BIST”) and the normal drive interrupt bit needs to remain asserted (see Section 9.5.2, “Miscellaneous Local Control”)

It is important to note that if the Tigon is not actively driving the bus as a bus master, then this output will reflect address bit-31 for the next bus master operation. If the Tigon is actively driving the bus as a bus master, then this output reflects address bit-31 for the current operation.

5.4 Configuration

The PCI interface of the Tigon requires system configuration at power-up just like any other PCI card. The Tigon will only respond to configuration commands until it is enabled to respond to as a target or allowed to perform bus master operations.

The first 64 bytes of the 256 byte configuration space are defined by the PCI specification. The remaining bytes are specific to the operation of the Tigon.

5.4.1 Access Methods

The 64 PCI defined configuration bytes can be accessed in two techniques by the Tigon. The first method is through the PCI required configuration space. Some of the registers are read-write while others are read-only through the configuration space. These same 64 bytes are also mapped as the first 64 bytes of the 16k byte shared memory target space. This provides the second method to access the registers. Registers which must be read-only from the configuration space are read-write when accessed through the target interface. This allows for fields such as Vendor_ID and Device_ID to be written and checked by diagnostics.

The configuration space must still be accessed first in order to enable the target interface. For implementation in which the Tigon will not really be on a PCI bus, the power-up firmware on the adapter can enable the target interface automatically if so desired. This only applies to custom adapters developed with the Tigon as the Ethernet interface.

5.4.2 Preloading from Serial EEPROM

The Tigon powers-up with the value of ‘h12AE as the Vendor_ID and ‘h0001 as the Device_ID. It always requests a 16k byte address range for the target shared memory. If no additional changes are required, then the power-up reset values can be used.

Any changes to the default values will be implemented by having the internal processor execute a small piece of code from the Serial EEPROM on the adapter. This code will be automatically executed after the ROM completes and can load any configuration register with new values. This process would take place within about 50 milliseconds of the PCI reset being de-asserted depending on how efficiently the code was written. These new configuration values would then be used by the system BIOS to configure the adapter.

6 Ethernet Transmit Interface

The Tigon's transmit Ethernet interface fully complies with the IEEE 802.3 specifications and is kept up to date with the 802.3z Gigabit Ethernet proposals.

The Tigon uses Ethernet Descriptors to keep track of packets being sent to the serial Ethernet interface. The format of these descriptors is fixed in order to allow the hardware to directly reference the fields within the descriptors.

The Ethernet transmit interface is responsible for sending packets to the external network interface by reading the associated transmit descriptor and the packet from the local memory buffer. Error conditions are monitored during the packet transmission and reported to the software through an event at the moment in which they occur. For the transmit interface, errors are considered unusual and are therefore treated as rare events.

Packets are sent only when a valid descriptor is ready and the transmit buffer pointers indicate the data is available.

6.1 Transmit Descriptors

The Ethernet Transmit Descriptors are set up by software in order to indicate to the transmit hardware where the packets to transmit are located. In a similar fashion to that used to communicate with the host, the descriptors are organized into a ring with a producer and a consumer index. After software initializes the fields within the descriptor and the first doubleword located at the Starting_Address, it updates the producer index indicating to the hardware that it can read the descriptor and establish any connection required. Whenever the producer and consumer indexes are equal there are no packets to transmit.

Each of the Transmit Descriptors are eight bytes in length as illustrated by Figure 3. The first 32-bits contain the byte address at which the packet starts within the external SRAM associated with the Tigon's buffer memory. Since all Ethernet packets must start on a doubleword boundary, the least significant three bits of the address should always be zero. All reserved bit positions are not referenced by the Tigon and can be used by firmware to store state information if so desired.

If while sequencing through the buffer reading the packet for transmission, the last word fetched was from the top of the transmit buffer space, the buffer consumer pointer will automatically be set back to the transmit buffer base address.

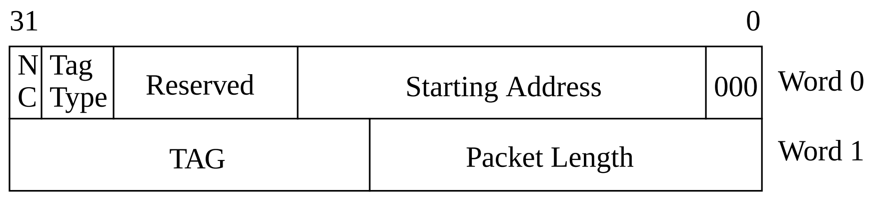

Figure 3. Ethernet Transmit Descriptor

Table 13. EthernetTransmit Descriptor Fields

	Bit	Field	Description
Word 0	31*	Don't Append CRC	This packet should be sent as is with no padding nor CRC added
	31-30*	Tag Type	00 = No tag operation 01 = Remove tag from packet 10 = Add tag to packet 11 = Replace tag in packet
	29-16	Reserved	
	20-0	Starting_Address	Byte address of start of packet
Word 1	31†	Don't Append CRC	This packet should be sent as is with no padding nor CRC added
	31-16*	Tag	Optional 16-bit VLAN Tag value
	15-0‡	Packet_Length	Length of packet in bytes 0 is not a legal value

* Tigon revision 5 only.

† Tigon revision 4 only.

‡ Tigon revision 4 only supports bits 14-0. Thus maximum packet is 32k - 8 bytes.

The lower 32-bits contain the length of the portion of this packet in bytes which is described by this descriptor. A single Transmit Descriptor is only capable of sending packets up to 64K - 8 bytes.

6.2 Transmit Descriptor Indexes

The Ethernet transmit descriptors are organized in a ring of descriptors located in a consecutive block of memory. The offsets into this block of memory which point to specific descriptors are called "Indexes." The two indexes which are used to maintain transmit descriptors are the producer index and the consumer index. The producer index is updated by firmware after the respective Transmit Descriptor fields have been initialized. The consumer index is updated by the hardware after it transmits the packet corresponding to that descriptor. Whenever the two indexes are equal, there are no additional packets to transmit.

Both the transmit producer and consumer indexes are 8 bits in length; therefore there are a total of 256 transmit descriptors respectively. One transmit descriptor must remain unused so that the producer doesn't overtake the consumer. This means that there are 255 usable transmit descriptors at any one time. Separate address decodes are provided which allow software to read or write either of these two indexes. For normal operations however, software would write the producer index and read the consumer index.

To facilitate the generation of internal events when the hardware has completed the transmission of a packet, a reference index register was created inside in the hardware. This reference index is the same size as the producer and consumer indexes. If at any point the reference index is not between the producer and the consumer nor equal to the producer, then an internal event is generated. Software can now set up the reference index to the current packet, knowing that an event will be generated when the hardware advances beyond that packet.

The format of the three index registers is shown in Figure 4. Only the Index field is read/write to the software. The Transmit Descriptor Base Address is read-only and is provided by the hardware so that reading these registers provides a complete address into the SRAM at which the Transmit Descriptor is located.

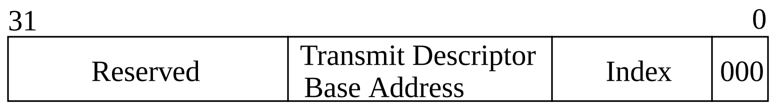

Figure 4. Ethernet Transmit Index

Table 14. Ethernet Transmit Index

Bits	Field	Description
31-21	Reserved	Always 0
20-11	T.D. Base Address	SRAM address of start of Transmit Descriptors
10-0	Index Offset Address	Byte offset address of descriptor entry

6.3 Transmit Descriptor Location

Since there are 256 transmit descriptors, they occupy either 2kBytes of local memory. The transmit buffers are located in the block of memory directly below (i.e. lower address) the receive descriptors. For reference, the receive descriptors are located directly below the DMA descriptors, which is below the transmit buffer. The Transmit Buffer Base register is used to define the address at which the transmit buffer starts.

6.4 Transmit Buffer Indexes

The Ethernet transmit buffer is organized as a circular buffer ring occupying a consecutive block of memory. The offsets into this block of memory which point to specific addresses are called “Indexes.” The two indexes which are used to maintain transmit buffer are the transmit buffer producer index and the transmit buffer consumer index. The producer index is updated by firmware to point to the next doubleword after the valid data to be sent. The consumer index is updated by the hardware as it transmits the packet to indicate where to fetch the next transmit data. Whenever the two indexes are equal, the transmit buffer is empty.

Transmit buffer indexes need only point to doubleword quantities, therefore the lower 3-bits are always zero. Otherwise they operate as just local memory pointers.

6.5 Transmit Flow Control

The Ethernet transmit interface supports the 802.3x flow-control mechanism in hardware. This feature must first be enabled in order to allow the hardware to send such flow control packets. Transmission of a valid 802.3x packet is done based on high and low water marks for number of receive descriptors unused and receive buffer space which is unused. These threshold are described in Chapter 9.11.6, “Ethernet Thresholds.” Once either high water mark has been exceeded, an “XOFF” packet will be sent as the next packet. If the “XOFF” length is about to expire, another flow control packet will be sent to refresh it up until the low resource drops below the low water mark at which point an “XON” message will be sent.

The following is the structure of the transmitted packet.

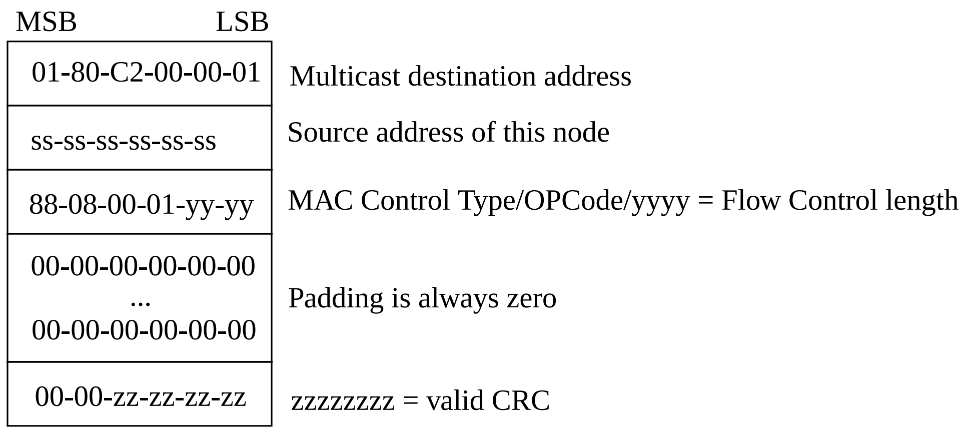

Figure 5. Transmit Flow Control Packet Structure

6.6 Transmit TAGs

The Ethernet transmit interface supports the 802.1Q Type-based TAGs. These 4-Byte TAGs exists between the source MAC address and the original Type field entry. Using the control bits which exist in the Transmit Descriptor the following operations are supported:

Table 15. Transmit TAG Functions

TAG Operations	Function
No TAG operation	Packet is unchanged.
Remove TAG	After source address 4-bytes are removed.
Add TAG	After source address 2-bytes are inserted from TAG Type register, and 2-bytes are inserted from TAG field in transmit descriptor.
Replace TAG	After source address replace next 2-bytes with TAG Type register, and replace next 2-bytes with TAG field in transmit descriptor.

6.7 Ethernet CRC Calculation

The Tigon uses the standard 32-bit CRC required by the Ethernet specification as its FCS in all packets. It also uses the same CRC during receive as the hash function for multicast filtering, and to verify the integrity of areas of the external non-volatile memory. The algorithm to generate the CRC is described here for completeness.

The checksum is the 32-bit remainder of the polynomial division of the data taken as a bit stream of polynomial coefficients, and a predefined constant which also represents binary polynomial coefficients. The checksum is optionally appended most-significant bit first to a packet which is to be sent down the wire. At the receiving side the division is repeated on the entire packet including the CRC checksum. The remainder is compared to a known constant. For details on the mathematical basis for CRC checksums, see Tanenbaum's *Computer Networks*, Third Edition, c1996. The 32-bit CRC polynomial divisor is shown below:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

6.7.1 Generating CRC

The following steps describe a method to calculate the CRC with the resulting 32-bit quantity having reversed bit order (i.e. most-significant bit x^{31} of the remainder is right-most bit). The data should be treated as a stream of bytes.

- Set remainder to 0xFFFFFFFF
- For each bit of data starting with least-significant bit of each byte:
  - If right-most bit (bit-0) of the current remainder XOR'd with the data bit equal 1, then remainder = (remainder shifted right 1 bit) XOR 0xEDB88320, else remainder = (remainder shifted right 1 bit).
- Invert remainder such that remainder = ~remainder.
- Remainder is CRC checksum.
- Right-most byte is the most-significant and is to be sent first.
  Swap bytes of CRC if big-endian byte ordering is desired.

6.7.2 Checking CRC

The following steps describe a method to check a stream of bytes which has a CRC appended.

- Set remainder to 0xFFFFFFFF
- For each bit of data starting with least-significant bit of each byte:
  - If right-most bit (bit-0) of the current remainder XOR'd with the data bit equal 1, then remainder = (remainder shifted right 1 bit) XOR 0xEDB88320, else remainder = (remainder shifted right 1 bit).
- Remainder should equal magic value 0xDEBB20E3 if CRC is correct.

7 Ethernet Receive Interface

The Tigon's Ethernet receive interface fully complies with the IEEE 802.3 specification and is kept up to date with the 802.3z Gigabit Ethernet proposals.

The Tigon uses receive Descriptors to keep track of packets being received from the serial Ethernet interface. The format of these descriptors is fixed in order to allow the hardware to directly reference the fields within the descriptors.

The Ethernet receive interface is responsible for accepting packets from the external network interface and placing them in the local memory buffer along with an associated receive descriptor. All data being received from the Ethernet interface goes through a small synchronizing FIFO which converts the data to the internal clock frequency.

Error conditions are monitored during the packet reception and reported to the software through the status word located in the Ethernet descriptor. Serializer/Deserializer integrity errors are always reported directly to the internal processor. They may also be indicated in the final status word if a receive packet was in progress. This allows the processor to monitor the quality of the serial channel and perform any necessary error reporting to the host.

7.1 Packet Structure

The packet structure for received packets in the local buffer memory is shown in Figure 6. This structure will always start on a doubleword boundary. Starting with Tigon revision 5, this structure will not start within 40 Bytes of the highest memory address in the receive buffer. This guarantees that at least the first 32 Bytes of the packet data will be contiguous in the local memory.

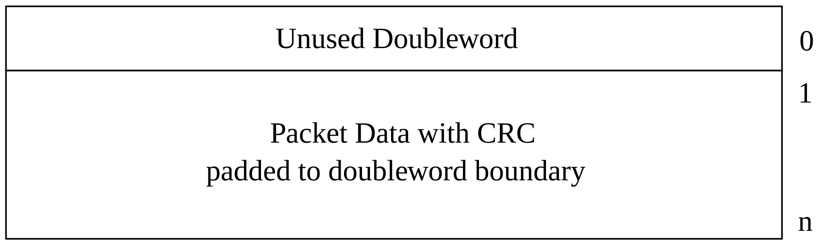

Figure 6. Ethernet Receive Memory Layout

7.2 Errors

The Ethernet receive interface monitors several indicators during the reception of a packet. Not all of these indicators can be considered fatal errors. In fact in some cases, such as diagnostics, they may be created intentionally. The hardware makes no attempt to do anything but report the information along with each packet. The only cases in which the packet is aborted is if the serial Ethernet interface indicates a fatal condition.

Errors which occur during a packet reception are reported in the status word field of the Receive Descriptor. Bits 31-16 indicate these error conditions. Every attempt has been made to report any condition which may help software determine the integrity of the Ethernet channel.

7.3 Receive Descriptors

The Ethernet Receive Descriptors are set up by hardware in order to indicate to the software where the received packets are located. In a similar fashion to that used to communicate with the host, the descriptors are organized into a ring with a producer and a consumer index. The hardware writes the fields within the descriptor only at the end of the packet. The hardware also updates the producer index indicating that software that it can now start processing the received packet. Whenever the producer and consumer indexes are equal there are no packets in the receive buffer.

Each of the receive descriptors are eight bytes in length as illustrated by Figure 7. All eight bytes of the descriptor are written at the end of the packet being received. The first 32-bits contain the byte address within the external SRAM associated with the Tigon’s receive buffer at which the packet starts. All received packets will be placed in the receive buffer starting at doubleword boundaries. All reserved bit positions are not used by the Tigon and will therefore be set to zero by the hardware.

The lower 32-bits of the Receive Descriptor contain the ending status information regarding the received packet and the final length.

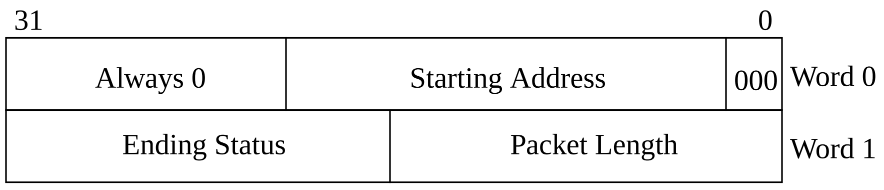

Figure 7. Ethernet Receive Descriptor

Table 16. Ethernet Receive Descriptor Fields

	Bit	Field	Description
Word 0	31-21	Reserved	Always 0
	20-0	Ending_Address	Byte address of start of next packet
Word 1	31-24	Reserved	Always 0
	23	Truncated Error	Packet truncated due to lack of buffer space
	22	Packet < 64 Bytes	Packet is less than 64 bytes
	21	Packet Aborted Error	Packet was aborted by MAC
	20	MII Nibble Error	Packet arrived with odd number of nibbles
	19	PHY Error	PHY detected a packet error
	18	Link Error	Link was deasserted during packet
	17	Collision Error	Packet contained a collision
	16	CRC Error	Packet failed CRC check
	15-0	Packet Length	Packet length in bytes (including CRC)

7.4 Receive Descriptor Indexes

The Ethernet receive descriptors are organized in a ring of descriptors located in a consecutive block of memory. The offsets into this block of memory which point to specific descriptors are called "Indexes." The two indexes which are used to maintain receive descriptors are the producer index and the consumer index. The producer index is updated by hardware after the respective Receive Descriptor fields have been initialized at the end of a packet. The consumer index is updated by the firmware after it completes the processing on the packet corresponding to that descriptor. Whenever the two indexes are equal, there are no packets in the receive buffer.

Both the receive producer and consumer indexes are 9 bits in length; therefore there are a total of 512 (i.e. 2^9) receive descriptors. The Receive Indexes point to 4-byte boundaries instead of the 8-byte boundaries of the Transmit Indexes. This enables the index to point to either word in the Receive Descriptor. The producer index advances to point to the second half of the descriptor when a new packet reception has started and the threshold number of bytes defined in Chapter 9.11.7, "Ethernet Receive State" register has been received. The producer index advances to point to the beginning of the next descriptor when the packet has been completely received. This feature is utilized as a mechanism for the hardware to indicate the difference between a packet reception in progress and one in which the entire packet has been received.

To facilitate the generation of internal events when the hardware has completed the reception of a packet, a reference index register was created inside in the hardware. This reference index is the same size as the producer and consumer indexes. If at any point the reference index is not between the producer and the consumer nor equal to the producer, then an internal event is generated. Software can now set up the reference index to the current packet, knowing that an event will be generated when the hardware advances beyond that packet.

The format of the three index registers is shown in Figure 8. Only the Index field is read/write to the software. The Receive Descriptor Base Address is read-only and is provided by the hardware so that reading these registers provides a complete address into the SRAM at which the Receive Descriptor is located. Bit-2 of the consumer index is fixed since that register does not need the ability to point to each half of a Receive Descriptor.

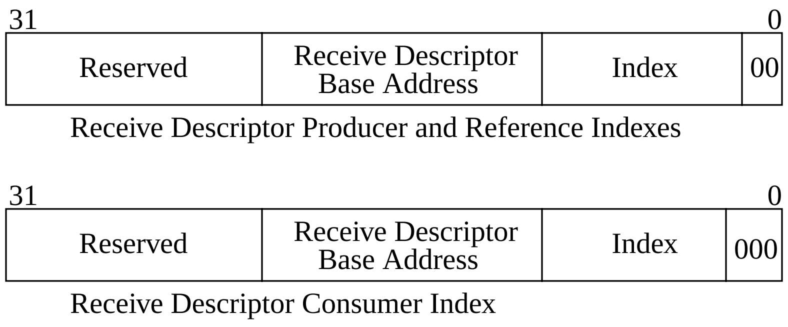
31
0
Reserved
Receive Descriptor Base Address
Index
00
Receive Descriptor Producer and Reference Indexes
31
0
Reserved
Receive Descriptor Base Address
Index
000
Receive Descriptor Consumer Index

Figure 8. Ethernet Receive Index

Table 17. Ethernet Receive Index

Bits	Field	Description
31-21	Reserved	Always 0
20-12	R.D. Base Address	SRAM address of start of Receive Descriptors
11-0	Index Offset Address	Byte offset address of descriptor entry

7.5 Receive Descriptor Location

There are 512 receive descriptors which occupy 4k Bytes of local memory. The receive descriptors are located in the block of memory directly below (i.e. lower address) the DMA descriptors, which is directly below the transmit buffer. The Transmit Buffer Base register is used to define the address at which the transmit buffer starts.

7.6 Receive Buffer Indexes

The Ethernet receive buffer is organized as a circular buffer ring occupying a consecutive block of memory. The offsets into this block of memory which point to specific addresses are called "Indexes." The two indexes which are used to maintain receive buffer are the receive buffer producer index and the receive buffer consumer index. The producer index is updated by the hardware as it receives the packet to indicate where it will place the next data received. The consumer index is updated by firmware to point to the first doubleword of the valid data it has yet to process. Whenever the two indexes are equal, the receive buffer is empty.

Receive buffer indexes only point to doubleword quantities, therefore the lower 3-bits are always zero. Otherwise they operate as just local memory pointers.

7.7 Receive Flow Control

The Ethernet receive interface supports the 802.3x flow-control mechanism in hardware. This feature must first be enabled in order to allow the hardware to monitor incoming packets and process the flow control packets. Reception of a valid 802.3x packet will update a counter which indicates how long the transmit interface should stop sending packets. Any packet which has already started transmission will not be affected. Both "XOFF" and "XON" packets are allowed.

The following is the structure of the expected packet. If the incoming packet does not match this structure, then the hardware will not do any processing and will place the packet in the receive buffer along with all non-flow control packets.

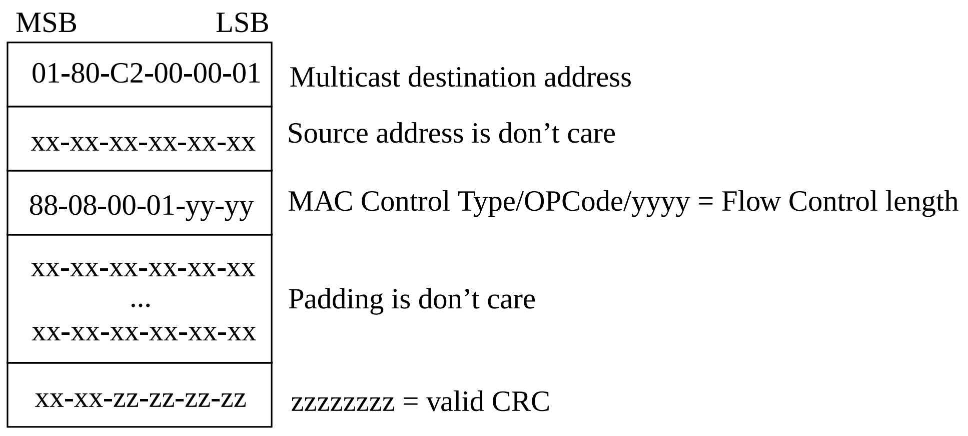

Figure 9. Receive Flow Control Packet Structure

8 DMA Assist

The Tigon contains two DMA channels to transfer data between its local memory and the host memory. Managing these channels in order to keep them active can require significant resources of the internal processor. Perhaps more important that the number of processor cycles which would be used is the fact that the processor would need to be very responsive to the needs of the DMA channels. To help off-load some of these tasks from the internal processor, the “DMA Assist” state machine was created to perform the most time critical tasks.

The DMA Assist logic primarily manages the task of restarting a DMA channel once it has completed with its prior operation. If the DMA channel is always kept busy, then maximum host memory bandwidth can be achieved. DMA descriptors were created in order for firmware to pass the relevant information about a required DMA to the Assist logic. These DMA descriptors reside in a small portion of the local memory and are organized into a ring structure.

8.1 High and Low Priority Rings

Each DMA channel is assigned two DMA descriptor rings. The Assist logic uses a descriptor off of one of the two rings to start a DMA operation. Any time there is a descriptor ready on the high priority ring, it will get serviced prior to any descriptors on the low priority ring. Once a DMA has started it will run until completion and will not be interrupted even if a higher priority descriptor has been queued.

It is anticipated that the low priority ring will be used to queue packet data transfers, while the high priority ring will be used to queue driver control messages. In this way important control information, which are usually small blocks of memory, can be completed quickly before the next large packet data transfer.

8.2 DMA Descriptor Location

The location of the DMA descriptors is determined by the Transmit Base register. The start of the descriptor region will always be 4k bytes less than the Transmit Base register. The Assist region starts with the DMA Read high priority ring. The remaining three rings follow immediately after each other such that the following order is established:

- DMA Write Low Priority (high memory)
- DMA Write High Priority
- DMA Read Low Priority
- DMA Read High Priority (low memory)

8.3 Combining DMA Descriptors

In applications where having a larger number of descriptors is more important than having high and low priority rings, it is possible to combine the four rings into two rings. When the configuration bit in the assist state register has been set to indicate this mode of operation. Software should only use the high priority ring registers and should disable any events from the low priority ring. Physically the new combined high priority ring uses the memory locations allocated to both the high and low priority rings prior to combining both of those descriptor rings.

8.4 Descriptor Pointers

The DMA descriptor rings are managed in the same fashion as the Ethernet transmit descriptor ring. The producer register is updated by firmware whenever a new entry is added to the ring. The consumer register is updated by the Tigon whenever a DMA has successfully completed. The reference register is updated by the firmware as a means to indicate when a DMA Assist event should be generated. See Chapter 4, “Internal Events,” for the details on how to use the reference register. At any given time, only 31 of the 32 possible DMA descriptors for each ring can be valid. This is done so that the pointers need not have the ability to distinguish between an empty and completely full ring.

8.5 Long DMA Descriptor

The Tigon should be configured for long DMA descriptors if using a host address greater than 32 address bits, or wishes to use hardware checksum logic. The long DMA descriptor format can support up to 64 host address bits. The descriptor format is shown in Figure 10.

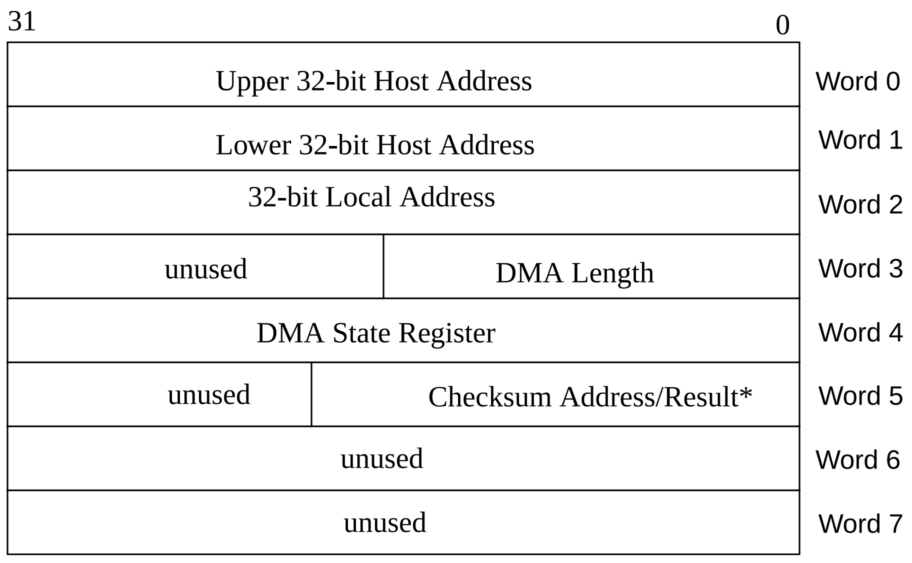

Figure 10. Long DMA Descriptor Format

* After the successful completion of a DMA, if the save checksum bit is set in the State register, then the 16-bit checksum will be saved in local memory. Tigon revision 4 stores the checksum into the 16-bit Checksum Result field, while starting with Tigon revision 5 the Checksum Address field contains an address of where in local memory to store the checksum result. In order to save the final checksum a local memory half-word write operation will be performed to any address at least half-word aligned.

All of the address and length fields in the DMA descriptor support byte values. This means a memory transfer can be performed between any two byte addresses for any number of bytes up to 64k Bytes. The first 5 fields will be to the appropriate Tigon register in order starting with Word 0. This process involves three doubleword reads from local memory in order to fetch the DMA descriptor.

The unused words are available for firmware to utilize as it desires. This area will most likely store state information for the firmware to reference once the DMA has completed.

The firmware defined words are available for software to utilize as it sees fit. This area will most likely store state information for the firmware to reference once the DMA has completed

8.6 Mini DMA Descriptor

The Tigon should can be configured for mini DMA descriptors if the host is using 32 or less address bits over the PCI bus and will not be saving the resulting checksum. The mini DMA descriptor format can only support up to 32 host address bits. The descriptor format is shown below.

The mini DMA descriptor was implemented to reduce the number of local memory read operations were necessary to read the descriptor in order to restart the DMA channel and maximize the number of available descriptors.

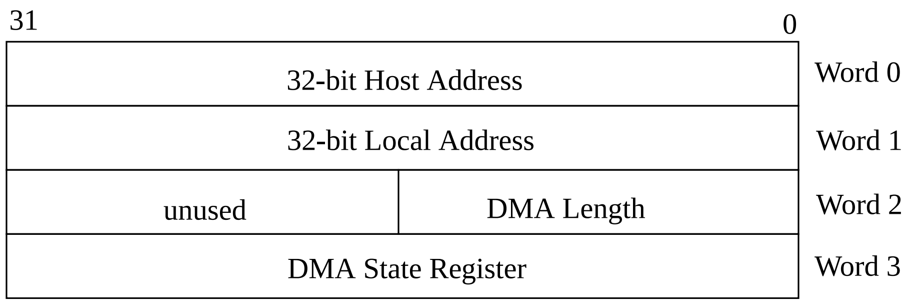

Figure 11. Mini DMA Descriptor Format

All of the address and length fields in the DMA descriptor support byte values. This means a memory transfer can be performed between any two byte addresses for any number of bytes up to 64k Bytes. The first 4 fields will be to the appropriate Tigon register in order starting with Word 0. This process involves two doubleword reads from local memory in order to fetch the DMA descriptor. The firmware should initialize the upper 32-bits of the host memory register to zero during initialization time in order to indicate that those bits are not being used in the system.

9 Register Descriptions

This chapter describes the registers which are accessible in the Tigon. Not all of the registers are inside the chip, some of them are stored in the external local memory of adapter. This technique allows for various non-critical state and control information to be stored in relatively inexpensive memory.

9.1 Host Register Access Methods

Registers are accessible using one of three methods as shown in Table 18. The host can access registers through the PCI's configuration space as well as through the Tigon's 16k Byte shared memory region. The internal processor accesses all registers as memory mapped locations within its memory space.

Table 18. Host Register Access Methods

Access Method	Region Size	Region Base Address
PCI Configuration	256 Bytes	Predetermined by PCI specification
PCI Shared Memory	16k Bytes	Set during system PCI configuration
PCI ROM	32k Bytes	Set during system PCI configuration

The first 256 Bytes of both methods accesses essentially the same registers. The only difference is that a few of the registers predefined by PCI specification are read-only when accessed in the PCI Configuration space and are otherwise read-write.

9.2 Shared-Memory Register Regions

The shared-memory region can be divided into several categories as shown in Table 19. Each of these categories will be described in a separate section of this chapter. All offset are relative to the beginning of the shared-memory region. Writes to all offsets which are labeled reserved will have no effect, while reads to these reserved locations will provide unpredictable results.

Table 19. Shared-Memory Categories

Host Offset	Category	Size in Bytes
000h-03Fh	PCI Configuration	64
040h-07Fh	General Control	64
080h-0BFh	Host DMA Control	64
0C0h-0FFh	Local Memory Configuration	64
100h-13Fh	Host DMA Assist Control	64
140h-17Fh	CPU A Control Registers	64
180h-1FFh	CPU A Internal Registers	128
200h-23Fh	MAC Registers	64
240h-27Fh	CPU B Control Registers	64
280h-2FFh	CPU B Internal Registers	128
300h-3FFh	Reserved	256
400h-47Fh	General Communications	128
480h-4FFh	MAC Statistics	128
500h-5FFh	Mailboxes	256
600h-7FFh	General Communications	512
800h-FFFh	Local memory Window	2k
1000h-3FFFh	Host DMA FIFO Access	12k
	Total	**16k**

The shared-memory region is much larger than the other regions in order to allow more flexibility in how the host can access locations in the local memory external to the Tigon. First, an additional 1k Bytes of the local memory is mapped into the shared-memory region in order to allow the host and the internal processor to have common memory which is used for communications between the processors. Secondly, a 2k Byte *window* was created so that the host can map any location in the local memory into the shared-memory region. As a general rule, host software does not need to be aware of which registers are internal to the Tigon and which are implemented in external local memory.

9.3 Register Quick Reference

Table 20. Tigon Registers – Quick Reference

Offset	Register	
000h-003h	PCI Device ID / Vendor ID	page 67
004h-007h	PCI Status / Command	page 67
008h-00Bh	PCI Class Code / Revision ID	page 68
00Ch-00Fh	PCI BIST/Header_Type/Latency_Timer/Cache_Line_Size	page 69
010h-014h	PCI Shared Memory Base Address	page 69
014h-02Bh	-	
02Ch-02Fh	PCI SubSystem ID / SubSystem Vendor ID*	page 70
030h-033h	PCI ROM Base Address	page 70
034h-03Bh	-	
03Ch-03Fh	PCI Max_Lat/Min_Grant/Interrupt_Pin/Interrupt_Line	page 71
040h-043h	Miscellaneous Host Control	page 73
044h-047h	Miscellaneous Local Control	page 74
048h-04Bh	Semaphore A*	page 76
048h-04Bh	Semaphore B*	page 76
050h-053h	Miscellaneous Configuration	page 76
054h-057h	Timer (bits 31-0)	page 76
058h-05Bh	Timer Reference A	page 76
05Ch-05Fh	PCI State	page 77
060h-063h	Main Event A	page 78
064h-067h	Mailbox Event A	page 79
068h-06Bh	Window Base Address	page 79
06Ch-06Fh	Window Data (Host only)	page 80
070h-073h	Main Event B*	page 78
074h-077h	Mailbox Event B*	page 79
078h-07Bh	Timer Reference B*	page 76
07Ch-07Fh	Serial Data	page 80
080h-083h	DMA Write Host Address (bits 63-32)	page 82
084h-087h	DMA Write Host Address (bits 31-0)	page 82
088h-08Fh	-	
090h-093h	DMA Read Host Address (bits 63-32)	page 81
094h-097h	DMA Read Host Address (bits 31-0)	page 81
098h-09Bh	-	
09Ch-09Fh	DMA Read Length	page 85
0A0h-0A3h	DMA Write State	page 84
0A4h-0A7h	DMA Write Local Address	page 86
0A8h-0ABh	DMA Write TCP/IP Checksum	page 87
0ACh-0AFh	DMA Write Length	page 85

Table 20. Tigon Registers – Quick Reference (continued)

Offset	Register	
0B0h-0B3h	DMA Read State	page 82
0B4h-0B7h	DMA Read Local Address	page 86
0B8h-0B7h	DMA Read TCP/IP Checksum	page 86
0BCh-0BFh	-	
0C0h-0C3h	Receive Buffer Base	page 89
0C4h-0C7h	Receive Buffer Producer	page 89
0C8h-0CBh	Receive Buffer Consumer	page 89
0CCh-0CFh	Received Packet Start Address (Packet in progress)	page 89
0D0h-0D3h	Transmit Buffer Base	page 90
0D4h-0D7h	Transmit Buffer Producer	page 90
0D8h-0DBh	Transmit Buffer Consumer	page 90
0DCh-0DFh	Transmit Next Packet Starting Address*	page 90
0E0h-0E3h	Receive Descriptor Producer	page 90
0E4h-0E7h	Receive Descriptor Consumer	page 91
0E8h-0EBh	Receive Descriptor Reference	page 91
0ECh-0EFh	CPU Priority	page 91
0F0h-0F3h	Transmit Descriptor Producer	page 92
0F4h-0F7h	Transmit Descriptor Consumer	page 92
0F8h-0FBh	Transmit Descriptor Reference	page 93
0FCh-0FFh	Transmit Next Descriptor Available*	page 93
100h-103h	Read Channel High Priority Producer	page 97
104h-107h	Read Channel High Priority Consumer	page 97
108h-10Bh	Read Channel High Priority Reference	page 97
10Ch-10Fh	Read Channel High Next Descriptor Available*	page 97
110h-113h	Read Channel Low Priority Producer	page 97
114h-117h	Read Channel Low Priority Consumer	page 97
118h-11Bh	Read Channel Low Priority Reference	page 97
11Ch-11Fh	Assist State	page 96
120h-123h	Write Channel High Priority Producer	page 97
124h-127h	Write Channel High Priority Consumer	page 97
128h-12Bh	Write Channel High Priority Reference	page 97
12Ch-12Fh	Write Channel High Next Descriptor Available*	page 97
130h-133h	Write Channel Low Priority Producer	page 97
134h-137h	Write Channel low Priority Consumer	page 97
138h-13Bh	Write Channel Low Priority Reference	page 97
13Ch-13Fh	-	

Table 20. Tigon Registers – Quick Reference (continued)

Offset	Register	
140h-143h	CPU A State	page 98
144h-147h	CPU A Program Counter (PC)	page 99
148h-14Bh	CPU A Hardware Breakpoint	page 99
14Ch-14Fh	CPU A Instruction*	page 100
150h-154h	CPU A Adjust Stack*	page 100
154h-157h	CPU A Internal SRAM Indirect Address	page 100
158h-15Bh	CPU A Internal SRAM Indirect Data	page 100
15Ch-15Fh	CPU A Watchdog Clear	page 101
160h-17Fh	-	
180h-1FFh	CPU A General Purpose Registers 0 - 31	page 101
200h-203h	MAC Transmit State	page 103
204h-207h	MAC Transmit Auto-Negotiation	page 104
208h-20Bh	MAC High Address	page 104
20Ch-20Fh	MAC Low Address	page 104
210h-213h	MAC Random backoff	page 104
214h-217h	MAC Length Encoding	page 105
218h-21Bh	-	
21Ch-21Fh	MAC Threshold	page 105
220h-223h	MAC Receive State	page 106
224h-227h	MAC Receive Auto-Negotiation	page 107
228h-22Bh	MAC Multicast Filter 1	page 107
22Ch-22Fh	MAC Multicast Filter 2*	page 107
230h-233h	MAC Multicast Filter 3*	page 107
234h-237h	MAC Multicast Filter 4*	page 107
228h-237h	SERDES Configuration*	page 107
238h-23Fh	PCS Configuration*	page 108
240h-243h	CPU B State*	page 99
244h-247h	CPU B Program Counter (PC)*	page 99
248h-24Bh	CPU B Hardware Breakpoint*	page 99
24Ch-24Fh	CPU B Instruction*	page 100
250h-254h	CPU B Adjust Stack*	page 100
254h-257h	CPU B Internal SRAM Indirect Address*	page 100
258h-25Bh	CPU B Internal SRAM Indirect Data*	page 100
25Ch-25Fh	CPU B Watchdog Clear*	page 101
260h-27Fh	-	
280h-2FFh	CPU B General Purpose Registers 0 - 31*	page 101

* First implemented in Tigon revision 5.

9.4 PCI Configuration Registers

The following describes the registers which are required by the PCI specification for configuration. Access to these registers can be obtained through either the PCI configuration address space, or through the Base0 shared-memory region of the Tigon adapter. Some register must be defined as read-only in the PCI configuration address space, but are allowed to be read-write when accessed by other means. More detailed description of each register can be obtained from the PCI Bus Specification. All reserved fields in the configuration region should are implemented as NO-OPs. When read they return a '0' value.

Table 21. PCI Configuration Registers

Register	PCI Config. Region	Local Access
Vendor ID	R/O	R/W
Device ID	R/O	R/W
Command	R/W	R/W
Status	R/W	R/W
Revision ID	R/O	R/W
Class Code	R/O	R/W
Cache Line Size	R/O	R/O
Latency Timer	R/W	R/W
Header Type	R/O	R/O
BIST	R/O	R/O
Base Address Reg 0	R/W	R/W
Base Address Reg 1-5 (not implemented)	-	-
SubSystem ID	R/O	R/W
SubSystem Vendor ID	R/O	R/W
Expansion ROM Base Address	R/W	R/W
Reserved	-	-
Int Line	R/W	R/W
Int Pin	R/O	R/O
Min. Grant	R/O	R/W
Max. Latency	R/O	R/W

9.4.1 Vendor ID

The 16-bit Vendor ID register identifies the manufacturer of the PCI adapter. Valid vendor identifiers are allocated by the PCI SIG to ensure uniqueness. This register is loaded after reset from the EEPROM. Alteon’s vendor ID is 0x12AE.

9.4.2 Device ID

The 16-bit Device ID register identifies the particular adapter within those made by the same manufacturer. A simple approach may be to set this field to the lower portion of the serial number or to identify the product type with this field by making it static for each product. This register is loaded after reset from the EEPROM. Default value after a reset is 0x0001.

9.4.3 Command

This 16-bit read-write register is used by the PCI based host to enable various features of the Tigon chip. All of the bit positions are predefined by the PCI specification. Not all bits in this register are implemented. Below is a list of the defined bits:

Table 22. Command Register

Bit	Description	Comments
15-10	Reserved	Tigon sets these bits to ‘0’
9	Fast Back-to-Back Enable	Enables fast back-to-back transactions to different devices. Tigon will not generate back-to-back write operations despite the fact that this enable bit is implemented.
8	System Error Enable	Enables system error detection. Tigon will not report address parity errors until this bit is set and parity error detection is enabled.
7	Wait Cycle Control	Controls whether address/data stepping is done. Tigon does not do stepping, therefore this bit will remain ‘0’.
6	Parity Error Enable	Enables data parity error detection. Tigon will not report data parity errors until this bit is set.
5	VGA Palette Snoop	Enables palette snoop on VGA devices. Tigon does not support this bit, therefore this bit will remain ‘0’.
4	Memory Write and Invalidate	Enables use of Memory Write and Invalidate command. Tigon revision 5 is the first implementation to use this bit.
3	Special Cycles	Enables device to monitor Special Cycles operations. Tigon does not support Special Cycles, therefore this bit will remain ‘0’.
2	Bus Master	Enables Tigon to behave as a bus master. Tigon will not become a bus master until this bit is set.
1	Memory Space	Enables Memory space accesses. Tigon will not respond to Memory accesses until this bit is set.
0	I/O Space	Enables I/O space accesses. Tigon does not support I/O space, therefore this bit will remain ‘0’.

9.4.4 Status

The 16-bit read-write Status register is used to indicate status information on the Tigon chip for PCI Bus related events. All of the bit positions are predefined by the PCI specification. Not all bits in this register are implemented. Below is a list of the defined bits:

Table 23. Status Register

Bit	Description	Comments
15	Detected Parity Error	Indicates a data parity error was detected even if parity reporting was not enabled. Tigon fully supports this bit.
14	Signaled System Error	Indicates this device asserted system error (SERR#). Tigon fully supports this bit.
13	Received Master Abort	Indicates this device was bus master and transaction was terminated with master-abort. Tigon fully supports this bit.
12	Received Target Abort	Indicates this device was bus master and received a target-abort. Tigon fully supports this bit.
11	Signaled Target Abort	Indicates this device initiated a target-abort. This bit is only set if an external master disappears during a target operation to the Tigon.
10-9	DEVSEL Timing	These bits encode the slowest timing of DEVSEL# except for configuration cycles. Valid entries are 00b for fast, 01b for medium, and 10b for slow. Tigon is capable of the medium timing.
8	Data Parity Error Detected	Indicates that this device was a bus master when a parity error was detected and reporting of parity errors is enabled. Tigon is capable of operating with this bit set.
7	Fast Back-to-Back Capable	Indicates whether fast back-to-back transactions can be accepted when transactions are not to the same agent. Tigon is capable of operating with this bit set.
6	UDF Supported	Indicates whether User Definable Features are implemented. The Tigon is capable of operating with this bit set.
5	66 MHz Capable	Indicates whether this device can operate with a 66 MHz PCI bus. All Tigon’s are capable of setting this bit, however Tigon revision 5 is the first design which can operate at speeds up to 66 MHz.
4-0	Reserved	Tigon sets these bits to ‘0’.

To clear a particular bit position, a write must be performed with that bit being a '1' and all other bit positions being a '0' to the Status. In this manner bits can be set by hardware conditions, and cleared by software.

9.4.5 Revision ID

The 8-bit Revision ID register is used by the manufacturer to identify the specific revision number of this adapter. Any value is allowable. This field should be viewed as an extension to the Device ID register. This field is fully programmable starting in Tigon revision 5. Prior versions always set this field to 0x01. It is recommended that this field be initialized to the board revision level.

9.4.6 Class Code

The 24-bit Class Code register identifies the generic function of the device. All of the legal values are specific in the PCI specification. This field is always set by hardware to the class code for an Ethernet interface (i.e. 0x020000).

9.4.7 Cache Line Size

The 8-bit read-write Cache Line Size register is used by the PCI based host to indicate to the Tigon chip the size of a cache line in 32-bit increments. This field is used to determine when the Memory Write and Invalidate command can be utilized. If this register is a zero, then the Memory Write and Invalidate command will not be used by the Tigon chip. This register is set to '0' at reset.

9.4.8 Latency Timer

The 8-bit read-write Latency Timer register is used by the PCI based host to indicate to the Tigon chip the number of PCI Bus clocks in which the Tigon may own the bus before checking to see if the bus should be relinquished. Only the upper 5-bits are usable as the lower 3-bits are hardwired to '0'. This register is set to '0' at reset.

9.4.9 Header Type

The 8-bit read-only Header Type register identifies both the layout of bytes 10h through 3Fh of the Configuration space, as well as whether this adapter contains multiple functions. If bit 7 is high it indicates a multi-function device, otherwise a single function device is selected. The remaining bits are all defined to be zero. This register is always '0' in the Tigon chip.

9.4.10 BIST

This 8-bit read-write register is used to initiate and report the results of any Built In Self Test. This field is fully programmable starting in Tigon revision 5. Prior versions always set this field to '0.'

The BIST register is also used as a means to enable a few special modes within the Tigon. It is not expected that this in any way restricts the normal usage of the BIST register since that special encodings are considered illegal according to the PCI specification and would only be used in proprietary PCI implementations. Table 24 indicates these special values.

Table 24. Special BIST Encodings

BIST Value	Description
0x7D	Enable Ethernet internal SERDES test outputs. The GigaBlaze outputs txclk, rxclk, syncdet, rxlockr, txlockr, passn are multiplexed onto the TDat[5:0] output pins respectively.
0x7E	Enable proprietary PCI arbitration output. PCI Address bit-31 of the active DMA channel is output on the PIntA output pin. See Chapter 5.3.5, “Additional PCI Arbitration Output” for additional details.

9.4.11 Base Address Register 0

This 32-bit read-write registers are used to establish the memory and I/O space the adapter requires within that of the system. Once the system has determined the needs of all of the adapters, the operating system can get booted.

The Tigon supports one Base Address Registers which must be located in the host’s memory space. The other five Base Address Registers are not implemented and will return a low in all bit positions when read.

Base Address Register 0 is required for the Tigon to function. It provides a 16k Byte control region from which the host can access the adapter. The layout of this register is shown in Table 25.

Table 25. Base Address Register 0

Bit	Description	Comments
31-14	Base Address	Upper addresses which are don't care should contain a '1'. Lower addresses which are part of the window should contain a '0'. See PCI specification.
13-4	Always 0	
3	Prefetchable	Indicates that there are no side effects on reads, the device returns all bytes regardless of byte enables, and processor writes can get merged. Tigon requires this bit to be not set.
2-1	Type	Encoded with the following values: 00b – located anywhere in 32-bit address space 01b – located below 1 MByte 10b – located anywhere in 64-bit address space 11b – reserved The Tigon supports only Type 00b
0	Memory Space	Must be '0'

9.4.12 ROM Base Address Register

This 32-bit read-write registers are used to establish the location of a 32k Byte ROM region within the host's memory space. This ROM region can be used for openboot support. This ROM region maps to the local memory space 8000h - FFFFh.

Table 26. ROM Address Register

Bit	Description	Comments
31-15	ROM Base Address	Upper addresses which are don't care should contain a '1'. Lower addresses which are part of the window should contain a '0'. See PCI specification.
14-1	Always 0	
0	Address Decode Enable	Set to a '1' to enable the use of this ROM region.

9.4.13 SubSystem ID

The 16-bit read-write SubSystem ID register is used by the board manufacturer for identification. Its usage is outside the scope of this document. This register is first implemented in Tigon revision 5.

9.4.14 SubSystem Vendor ID

The 16-bit read-write SubSystem Vendor ID register is used by the board manufacturer for identification. Its usage is outside the scope of this document. This register is first implemented in Tigon revision 5.

9.4.15 Interrupt Line

The 8-bit read-write Interrupt Line register is used to communicate interrupt line routing information. This field is set after configuration by the host and later used by any driver which needs to know which physical interrupt on the system interrupt controller is assigned to this device. Values in this register are system architecture specific.

9.4.16 Interrupt Pin

The 8-bit Interrupt Pin register is used to indicate which interrupt pin the device uses. A value of 01h corresponds to INTA#. The Tigon always returns the value 01h.

9.4.17 Minimum Grant

The 8-bit Minimum Grant register is used to indicate the device's desired minimum grant period in units of 250 ns assuming a PCI clock rate of 33 MHz. Devices should specify values that will allow them to most effectively use their internal resources as well as the PCI Bus. This register is always set to 0x40 in the Tigon.

9.4.18 Maximum Latency

The 8-bit Maximum Latency register is used to indicate the device's desired maximum time between being granted the PCI Bus in units of 250 ns assuming a PCI clock rate of 33 MHz. Devices should specify values that will allow them to most effectively use their internal resources. The Tigon hardware always set this register to '0' to indicate the system default will be acceptable.

9.5 General Control Registers

The General Control Registers control the overall behavior of the Tigon. The majority of the configuration bits are found within this group of registers. Table 27 shows the offsets for each of the registers in this group.

Table 27. General Control Registers

Offset	Register	Access
040h-043h	Miscellaneous Host Control	R/W
044h-047h	Miscellaneous Local Control	R/W
048h-04Bh	Semaphore A	R/W
04Ch-04Fh	Semaphore B	R/W
050h-053h	Miscellaneous Configuration	R/W
054h-057h	Timer	R/W
058h-05Bh	Timer Reference A	R/W
05Ch-05Fh	PCI State	R/W
060h-063h	Main Event A	R/W
064h-067h	Mailbox Event A	R/W
068h-06Bh	Window Base Address	R/W
06Ch-06Fh	Window Data (Host only)	R/W
070h-073h	Main Event B	R/W
074h-077h	Mailbox Event B	R/W
078h-07Bh	Timer Reference B	R/W
07Ch-07Fh	Serial UART Data	R/W

9.5.1 Miscellaneous Host Control

The Miscellaneous Host Control register is used to control various functions within the Tigon normally controllable from the host interface. Each bit has a separate function from any other bit in this register. All R/W bits are located in the bottom byte so that when the host writes to this register to set the "No Swap" bit, that the least significant byte can be replicated by the software into all byte positions. This will ensure that the register behaves as expected prior to this bit being set.

Table 28. Miscellaneous Host Control Register

Bits	Field	Description	Access
31-28	Gate Array Revision	'3' for Tigon Chip Express release. '4' for Tigon LSI gate array release. '5' for Tigon LSI standard cell release.	R/O
27-8	Reserved	Always 0	
7	Reserved	Always 0	
6	Mask PCI Interrupt Output*	Set bit to mask off PCI Interrupt outputs. When cleared, any pending interrupt will be generated.	R/W
5	Enable Endian Word Swap	Set bit to enable endian word swapping when accessing Tigon through target interface	R/W
4	Enable Endian Byte Swap	Set bit to enable endian byte swapping when accessing Tigon through target interface	R/W
3	Hard Reset	Set bit to force a complete hardware reset of everything except PCI configuration registers (offsets 0x00-0x3F) and PCI State register (offset 0x5c).	R/W
2	Reserved	Always 0	
1	Clear Interrupt	Write bit is clear host PCI interrupt IntA	W/O
0	Interrupt State	This bit reflects the state of the PCI IntA pin	R/O

* First implemented in Tigon revision 5.

9.5.2 Miscellaneous Local Control

The Miscellaneous Local Control register is used to control various functions within the Tigon. All bits are set to zero (i.e. disabled) during reset.

Table 29. Miscellaneous Local Control Register

Bits	Field	Description	Access
31-28	Reserved	Always 0	
27	PCI Interrupt*	PCI IntA asserted interrupt. Useful for detecting interrupts from external devices.	R/W
26	UART Receive FIFO overrun	Receive 2 byte FIFO has overrun, oldest data was dropped	R/W
25	UART Receive Ready	Receive has a byte available	R/W
24	UART Transmit Done	Transmit done sending byte, ready for another byte	R/W
23	Serial EEPROM Data input	Input from bi-directional serial EEPROM data pin	R/O
22	Serial EEPROM Data output	Value of data to drive out when output enabled	R/W
21	Serial EEPROM Data output enable	When asserted, Tigon drives the value of Serial EEPROM data output	R/W
20	Serial EEPROM Clock output	Directly controls the clock output pin	R/W
19	MII Management Data input	Input from bi-directional MII Management data pin	R/O
18	MII Management Data output	Value of data to drive out when output enabled	R/W
17	MII Management Data output enable	When asserted, Tigon drives the value of MII Management data output	R/W
16	MII Management Clock output	Directly controls the clock output pin	R/W
15-14	Misc. Pins [1:0] outputs	Outputs which are defined by board level design. Tigon revision 4 called these outputs LAddr[22:21].	R/W
13-12	Misc. Pins [1:0] output enables*	When asserted, Tigon drives Misc. Pin outputs.	R/W
11-10	Misc. Pins [1:0] inputs*	Input from bi-directional Misc. Pins.	R/W
9-8	SRAM Bank Enable†	Size of each SRAM bank: 00=disable, 01=1MByte, 10=512kByte, 11=256kByte	R/W
7	Misc_Out[2]	Outputs which is defined by board level design. Tigon revision 4 uses this output for PHY loopback.	R/W
6	Traffic LED	Controls the Traffic LED	R/W
5	Link LED	Controls the Link LED	R/W
4	Enable Flash Writes	Enables writing to Flash region	R/W
3	Fast Flash	Flash region is to use fast 1 cycle accesses	R/W
2	Set Interrupt	Write bit is set host PCI interrupt IntA	W/O
1	Clear Interrupt	Write bit is clear host PCI interrupt IntA	W/O
0	Interrupt State	This bit reflects the state of the PCI IntA pin	R/O

* First implemented in Tigon revision 5.

† SRAM Bank Enable=10 is first implemented in Tigon revision 5.

9.5.3 Miscellaneous Configuration

The Miscellaneous Config register is used as an extension to the Miscellaneous Local register. There are several fields used to control several small counters associated with the free-running 32-bit Timer inside the Tigon. The prescale function is performed on the clock prior to advancing the Timer register to provide a resolution as close as possible to 1 microsecond.

Table 30. Miscellaneous Configuration Register

Bits	Field	Description	Access
31-29	Reserved	Always 0	
28	Transmit Clock Output Enable	Tigon's transmit clock output enable.	R/W
27-26	Transmit Clock Select	Select the clock source for the transmit MAC. Must be set prior to resetting MAC every time these bits change. The internal SERDES can be used with all modes except MII. 00 - No ext. gig. SERDES attached - TClk_in=62.5 MHz 01 - MII selected - RClk0=2.5 or 25 MHz 10 - External SERDES attached - TClk_in=125 MHz 11 - GMII attached - TClk_in=125 MHz	R/W
25-24	Receive Clock Select	Select the clock source for the receive MAC. Must be set prior to resetting MAC every time these bits change. Reset value of this field is 11 which requires RClk1 input pin to have a clock source. 00 - Internal SERDES selected 01 - GMII selected - RClk1=125 MHz 10 - External SERDES selected - RClk1=62.5 MHz 11 - MII selected - RClk1=2.5 or 25 MHz	R/W
23-21	Reserved	Always 0	
20	Synchronous SRAM	Enable Synchronous SRAM timing.	R/W
19	Split Mailbox Events*	Split mailbox events such that lower 16 events go to Mailbox Event A and upper 16 events go to Mailbox Event B.	R/W
18	PCI Interrupt Input Enable*	Enable PCI Interrupt input attn (bit-27 of this register).	R/W
17	UART Serial Input*	Directly reflects state of serial input pin.	R/O
16	Run UART at local clock speed	If set, forces UART to send and receive each bit for only one local clock cycle.	R/W
15-8	UART Half-bit Timer	Number of timer advances which correspond to a half-bit time for the desired UART speed. If Timer advances every microsecond, use: 0x34 for 9600 baud 0x1d for 19200 baud 0x0d for 38400 baud	R/W
7	Reserved	Always 0	
6-0	Timer Prescaler	Local clock frequency in MHz, minus 1, which should correspond to each advance of the Timer. Example: For 48.0 MHz clock use 47 (0x2f). For 100.0 MHz clock use 99 (0x63).	R/W

* First implemented in Tigon revision 5.

9.5.4 Timer

The Timer register is a 32-bit counter implemented inside the Tigon. This counter is free-running and is used by the internal processor to keep track of relative time. The timer is associated with the local clock frequency. A 5-bit prescale function is performed on the clock prior to advancing the Timer register to provide a resolution as close as possible to 1 microsecond.

This register can be used in conjunction with the Timer Reference register to generate an event at a predetermine time. For example, if an event is desired in 50 microseconds, the current Timer value can be read and incremented by about 50. This value can be placed in the Timer Reference register so that an event can be generated at the appropriate time.

A queue of time-based events is managed by the internal processor so that many events can be queued for the future. As long as the queue is managed in an ordered fashion so that the top of the queue contains the time of the event closest to the current time, this top entry can be placed in the Timer Reference register.

This register is read-write only for initialization and diagnostic purposes. The internal processor writes zeros to initialize this register after the internal diagnostics complete.

9.5.5 Timer Reference A/B

The Timer Reference A and Timer Reference B registers are 32-bit registers used in conjunction with the Timer register to generate the Timer Event within the Tigon. These registers can contain any value, but must be written as a 32-bit quantity.

The hardware will set the Timer Event bit in the appropriate Main Event register as soon as the Timer and the Timer Reference register are equal. The event bit will remain asserted until a write is performed to the Timer Reference register.

! **NOTE:** *Software should verify prior to writing this register that the time about to be written has not already passed or is the current time. In the first case the hardware will miss the event until the Timer rolls over and comes back to the referenced time. In the second case the hardware may miss the event depending on when the Timer advances.*

9.5.6 Semaphore A/B

The Semaphore A and B registers allow access to both internal processors to a hardware semaphore mechanism. These registers are first implemented in Tigon revision 5. Writes to these registers of any data value indicate the desire to toggle the own/not-own states of the single semaphore bit. Reads of these registers will provide a ‘1’ if that register owns the semaphore, ‘0’ otherwise. To obtain the semaphore the normal operation is a loop of write ‘0’ then read until the read result is non-zero. To release the semaphore the normal operation is to write ‘0’.

9.5.7 PCI State

The PCI State register is used to control several functions within the Tigon associated with the PCI interface. All bits are set to zero (i.e. disabled) during reset. This register in not affected by the reset function defined in Chapter 9.5.1, "Miscellaneous Host Control" register.

Table 31. PCI State Register

Bits	Field	Description	Access
31-28	Default PCI Write Command	Use this command for all PCI write transactions. This field should normally be initialized to '7'.	R/W
27-24	Default PCI Read Command	Use this command for all PCI reads transactions of < 3 words. This field should normally be initialized to '6'.	R/W
23	Assert all PBEs on Writes*	During DMA Write operations, always drive all byte enables. This forces double-word alignment without having to adjust the DMA channel addresses and length.	R/W
22	PCI ROM Retry*	Force PCI retry operations for accesses to ROM region, if ROM region enabled.	R/W
21	PCI ROM Desired*	Enables PCI ROM Base Address register to be visible to the host.	R/W
20	32-bit PCI bus	Asserted if on a 32-bit PCI bus, otherwise indicates a 64-bit bus. Starting with Tigon revision 5, writes to this bit will force true 32-bit PCI operation even if host indicated 64-bit operation. Will cause PCI problems if this bit is asserted on a true 64-bit bus.	R/W
19	66 MHz PCI Bus*	Asserted if PCI bus is operating between 33 - 66 MHz, otherwise indicates operating between 0 - 33 MHz. Prior to Tigon revision 5, functioned like bit-18 except only for the write DMA channel.	R/O
18	No word swap DMA	Disable word swapping for all DMA transactions. Prior to Tigon revision 5, only affected read DMA channel.	R/W
17	Use Mem_Read_Multiple PCI Command	Use Mem_Read_Multiple command in place of Mem_Read_Line for DMA reads.	R/W
16	FIFO Retry Enable	Enable PCI retry response to PCI target accesses to FIFO when FIFO cannot complete operation. This bit is not used for normal operation.	R/W
15-8	Minimum DMA	Minimum number of PCI words each DMA channel is allowed to keep the PCI bus without allowing accesses by the other DMA channel. This guarantees a minimum PCI usage rather than the usual alternate per burst behavior.	R/W
7-5	DMA Write Maximum DMA	Encoded bits which force termination of PCI write operations at any of the following byte boundaries: disable, 4/8 (i.e. one bus width), 16, 32, 64, 128, 256, 1k.	R/W
4-2	DMA Read Maximum DMA	Encoded bits which force termination of PCI read operations at any of the following byte boundaries: disable, 4/8 (i.e. one bus width), 16, 32, 64, 128, 256, 1k.	R/W
1	Provide Length	Use non-standard PCI command which provide transfer length indication. See PCI Interface chapter.	R/W
0	Force Reset	Will force an immediate reset of the PCI interface. All state information will be lost. This bit is self clearing.	W/O

* First implemented in Tigon revision 5.

9.5.8 Main Event A/B

Table 32. Main Event Register

Bits	Field	Description
31	SW13	Software defined event (highest priority)
30	Serial UART	UART needs attention
29	SW12	Software defined event
28	Timer	Timer Reference reached
27	Remote CPU Attention*	Other CPU has signalled an attention. Clear by writing a ‘1’ to this bit position.
26	Ethernet Transmit Attention	Transmit needs attention
25	Ethernet Receive Attention	Receive needs attention
24	SW11	Software defined event
23	SW10	Software defined event
22	Mailbox	Mailbox Event asserted
21	SW9	Software defined event
20	DMA RD Attention	Read DMA stopped due to error
19	DMA WR Attention	Write DMA stopped due to error
18	DMA RD Completed	Read DMA completed successfully
17	DMA WR Completed	Write DMA completed successfully
16	SW8	Software defined event
15	Assist RD High Completed	Assist logic high priority read DMA descriptor completed successfully
14	Assist WR High Completed	Assist logic high priority write DMA descriptor completed successfully
13	Assist RD Low Completed	Assist logic low priority read DMA descriptor completed successfully
12	Assist WR Low Completed	Assist logic low priority write DMA descriptor completed successfully
11	SW7	Software defined event
10	Set Remote CPU Attention*	Write-only bit to set the Remote CPU Attention Event in the other Main Event register.
9	SW6	Software defined event
8	SW5	Software defined event
7	SW4	Software defined event
6	Ethernet Receive Completed	End of current packet has been received
5	Ethernet Receive Started	Start of next packet has been received
4	SW3	Software defined event
3	SW2	Software defined event
2	Ethernet Transmit Completed	Transmit packet(s) has completed
1	SW1	Software defined event
0	SW0	Software defined event (lowest priority)

* First implemented in Tigon revision 5. Prior revisions use these bits as software events.

The Main Event A and B registers are the primary methods which are used to alert the Tigon internal processors when action is required. Each bit in this register corresponds to a particular hardware or software state which should be read in the software event dispatching loop. The higher the bit number the more significant the event. Table 32 shows the bit ordering of this register.

9.5.9 Mailbox Event A/B

The Mailbox Event A and B registers are 32-bit registers with each bit corresponding to the Mailbox entry which was written by the host. For example, bit 0 represents Mailbox 0. Each mailbox consists of an 8-byte region. Only writes to the second half of this region will set a mailbox event. Writes to the mailboxes by the internal processor will not set these event bits. Mailbox Events can be disabled by writing a ‘1’ in the bit position of the bit which is to cleared.

If the Split Mailbox Events bit is not set in the Misc. Local register then these two mailbox event registers function identically. That is event will show up in both registers. If the Split Mailbox Event bit is set, then the lower 16 mailboxes will be associated with Mailbox Event A and the upper 16 mailbox events will be associated with Mailbox Event B. In all cases the mailbox event bit will still be in the same bit position as it would without split mailbox events, it is just that half of the mailboxes would not be accessible through each register. This allows firmware to organize the mailbox event depending on which processor is to be alerted. If only one processor needs mailbox events, then there is no reason to split the events -- the other processor need not reference its corresponding mailbox event register.

9.5.10 Window Base Address

The Window Base Address register defines the local memory address which is to be the base for the 2k Byte window provided by the Tigon. This register may contain any valid local memory address, but the usage of the least significant 11-bits varies depending on how the local memory is to be accessed. If the 2k Byte window is used, then the least significant 11-bits are ignored and are substituted with zeros. If the Window Data register is referenced, then the entire Window base Address register is used to indicate the local memory address of the operation. Figure 12 shows the layout of this register.

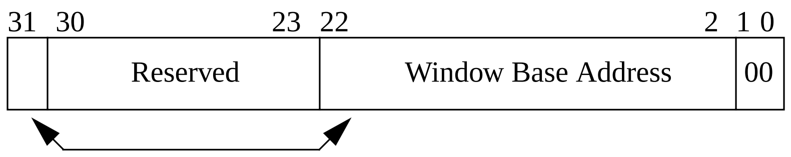

Figure 12. Window Base Address Register

Due to the arrangement of the Tigon’s local memory map, bit-31 is also considered a part of the Window Base Address register.

! **NOTE:** *The Window Base cannot be used by the host to access registers internal to the Tigon. It is only used to access memory external to the Tigon.*

9.5.11 Window Data

The Window Data register is normally used to access locations in the local memory when the actual 2k Byte local memory window provided by the Tigon is unavailable. Any access to this register is the same as an access to offset 0h of the local memory window. Only 32-bit operations are supported to this register.

This register combined with the Window Base Address register provide an indirect method to access the entire local memory address space.

9.5.12 Serial Data

The bottom byte of this register is used to provide data to or accept data from the UART. Any data written will immediately be sent out the UART. If the attention condition in the Misc. Local register indicates that there is data ready to be read, then the actual input byte is available by reading this register. The input FIFO is two bytes deep and will automatically remove the oldest data and generate an overflow attention if this serial data register is not read prior to the third byte arriving. Reads from the this register are destructive and automatically remove a byte from the input FIFO.

9.6 Host DMA Registers

The Host DMA Registers are used to control the two DMA channel which transfer data between host memory and the local memory of the adapter. The DMA Read hardware is completely independent to the DMA Write hardware. This means that transfers in both could be in process at the same time.

The two DMA channel are named after the PCI command which gets issued by each channel. That is, DMA Read refers to host memory reads by the adapter, and similarly DMA Write refers to writes to host memory by the adapter.

Table 33. Host DMA Registers

Offset	Register	Access
080h-083h	DMA Write Host Address (bits 63-32)	R/W
084h-087h	DMA Write Host Address (bits 31-0)	R/W
088h-08Fh	Reserved	
090h-093h	DMA Read Host Address (bits 63-32)	R/W
094h-097h	DMA Read Host Address (bits31-0)	R/W
098h-09Bh	Reserved	
09Ch-09Fh	DMA Read Length	R/W
0A0h-0A3h	DMA Write State	R/W
0A4h-0A7h	DMA Write Local Address	R/W
0A8h-0ABh	DMA Write TCP/IP Checksum	R/W
0ACh-0AFh	DMA Write Length	R/W
0B0h-0B3h	DMA Read State	R/W
0B4h-0B7h	DMA Read Local Address	R/W
0B8h-0BB7h	DMA Read TCP/IP Checksum	R/W
0BCh-0BFh	Reserved	

9.6.1 DMA Read Host Address

This is the 64-bit register/counter used to point to the next location in host memory where data is going to be read from. Only the least significant 32-bits are implemented as a counter, the upper 32-bits are merely a register. This means that any transfer which is to cross a 4GByte boundary, must be divided into more than one DMA operation.

If the upper 32-bits are all zero, then only a single PCI address cycle will be performed. In a 32-bit host address environment the upper 32-bits can be initialized to zero and then left alone, or starting with Tigon revision 5, if the Assist Mini descriptors are used, the upper 32-bits will automatically be set to zero during the load of each descriptor.

The initial value of the least significant 3-bits of this register are used along with the least significant 3-bits of the local address register to determine if any byte alignment operations will be necessary.

9.6.2 DMA Write Host Address

This is the 64-bit register/counter used to point to the next location in host memory where data is going to be written. Only the least significant 32-bits are implemented as a counter, the upper 32-bits are merely a register. This means that any transfer which is to cross a 4GByte boundary, must be divided into more than one DMA operation.

If the upper 32-bits are all zero, then only a single PCI address cycle will be performed. In a 32-bit host address environment the upper 32-bits can be initialized to zero and then left alone, or starting with Tigon revision 5, if the Assist Mini descriptors are used, the upper 32-bits will automatically be set to zero during the load of each descriptor.

The initial value of the least significant 3-bits of this register are used along with the least significant 3-bits of the local address register to determine if any byte alignment operations will be necessary.

9.6.3 DMA Read State

The DMA Read State register controls the operation of the Read DMA channel. If this register is to change, it must be written prior to the DMA Read Length register.

This DMA channel is activated whenever bit-2 is set to a high value. The hardware will automatically clear this bit when the DMA completes.

The checksum calculation can be disabled if it desirable to have the checksum logic hold the current value while a Read DMA operation is performed of data which is not to be included in the final checksum.

Normally all data is byte swapped in groups of four bytes as it is received from the PCI bus. This is done since the PCI bus is little endian and this adapter is naturally big endian. The No Swap bit in this register is used to indicate if this byte swap is to be disabled for this DMA operation. Word swapping can be disabled in the PCI State register but is normally left enabled.

The ability to reset the Read DMA logic is included to allow current DMA operations to be aborted. Any active PCI operation is immediately terminate gracefully before the FIFO is reset. This bit will is not intended to be used for normal operation of the DMA channel.

Table 34. DMA Read State Register

Bits	Field	Description	Access
31	FIFO Local Error	FIFO Error on local memory side of FIFO.	R/O
30	FIFO Underrun	FIFO experienced an underrun condition. Bit-31 indicates which side of FIFO detected the error.	R/O
29	FIFO Overrun	FIFO experienced an overrun condition. Bit-31 indicates which side of FIFO detected the error.	R/O
28	Reserved	Always 0	
27	Host Address Overflow Error	The Host address has incremented across a 32-bit address boundary.	R/O
26	PCI Parity Error	PCI parity error detected during host read operation.	R/O
25	PCI Master Abort	DMA channel generated a PCI master abort.	R/O
24	PCI Target Abort	DMA channel received a PCI target abort.	R/O
23	DMA Progress	This bit is set when the DMA channel writes any words to local memory. To clear this bit, read state register, clear bit-2, write result back to state register.	R/W
22-21	Reserved	Always 0	
20	No PCI bus Mastering	Block this DMA channel from performing any PCI bus master operations.	R/W
19	Invert Checksum*	During reads from checksum register, invert checksum.	R/W
18	Byte Bucket	Drop all data prior to writing to local memory.	R/W
17	Update Transmit Producer	When DMA is completed, the local address should be written to the transmit buffer producer register.	R/W
16-12	Empty Words	Indicates the number of empty 64-bit entries currently in FIFO up to a maximum of 31. Normal empty value is 0x1f.	R/O
11	Save Checksum	Indicates to Assist logic to save checksum after DMA has completed as long as Mini-Format descriptors are not being used.	R/W
10	Clear Checksum	Forces checksum to be cleared prior to the start of DMA.	W/O
9	Enable Checksum	Set bit to enable checksum calculation on data.	R/W
8-4	Threshold	Number of empty 64-bit entries in FIFO before DMA channel requests PCI bus operation. Must be a value between 1-31. Tigon revision 5 maps a value of 0 to 32 entries.	R/W
3	Force 32-bit PCI	Force PCI operation to be as if on a 32-bit PCI bus.	R/W
2	Active	Set bit to start DMA channel. Bit will automatically clear when DMA finishes.	R/W
1	No Swap	Set bit to disable endian byte swap on data from PCI bus.	R/W
0	Reset	Set bit to clear Read FIFO and Read DMA.	W/O

* First implemented in Tigon revision 5.

9.6.4 DMA Write State

The DMA Write State register controls the operation of the Write DMA channel. If this register is to change, it must be written prior to the DMA Write Length register.

Table 35. DMA Write State Register

Bits	Field	Description	Access
31	FIFO Local Error	FIFO Error on local memory side of FIFO.	R/O
30	FIFO Underrun	FIFO experienced an underrun condition. Bit-31 indicates which side of FIFO detected the error.	R/O
29	FIFO Overrun	FIFO experienced an overrun condition. Bit-31 indicates which side of FIFO detected the error.	R/O
28	Reserved	Always 0	
27	Host Address Overflow Error	Host address has incremented across a 32-bit boundary.	R/O
26	PCI Parity Error	PCI parity error detected by target during host write.	R/O
25	PCI Master Abort	DMA channel generated a PCI master abort.	R/O
24	PCI Target Abort	DMA channel received a PCI target abort.	R/O
23	DMA Progress	This bit is set when the DMA channel writes any words to local memory. To clear this bit, read state register, clear bit-2, write result back to state register.	R/W
22	Reserved	Always 0	
21	Set Interrupt*	Set Interrupt when this DMA completes.	R/W
20	No PCI bus Mastering	Block this DMA channel from performing any PCI bus master operations.	R/W
19	Update Receive Producer	Prior to DMA starting, update the receive producer to the local address should be written to the receive buffer.	R/W
18	Byte Bucket	Send all data to byte bucket and not to PCI interface.	R/W
17	Disable producer compare	Disable the default local memory DMA pause when DMA buffer read pointer reaches receive buffer producer.	R/W
16-12	Valid Words	Indicates the number of valid 64-bit entries currently in FIFO up to a maximum of 31. Normal empty value is '0'.	R/O
11	Save Checksum	Indicates to Assist logic to save checksum after DMA has completed with Long-Format descriptors.	R/W
10	Clear Checksum	Forces checksum to be cleared prior to the start of DMA.	W/O
9	Enable Checksum	Set bit to enable checksum calculation on data.	R/W
8-4	Threshold	Number of valid 64-bit entries in FIFO before DMA channel requests PCI bus operation. Once the last word to be written to the host has been placed into the FIFO, the host bus will be requested until the FIFO goes empty. Must be a value between 1-31. Tigon revision 5 maps a value of 0 to 32 entries.	R/W
3	Force 32-bit PCI	Force PCI operation to be as if on a 32-bit PCI bus.	R/W
2	Active	Set bit to start DMA channel. Bit will automatically clear when DMA finishes.	R/W
1	No Swap	Set bit to disable endian byte swap on data from PCI bus.	R/W
0	Reset	Set bit to clear Read FIFO and Read DMA.	W/O

* First implemented in Tigon revision 5.

This DMA channel is activated whenever bit-2 is set to a high value. The hardware will automatically clear this bit when the DMA completes.

The checksum calculation can be disabled if it desirable to have the checksum logic hold the current value while a Write DMA operation is performed of data which is not to be included in the final checksum.

Normally all data is byte swapped in groups of four bytes as it is sent to the PCI bus. This is done since the PCI bus is little endian and this adapter is naturally big endian. The No Swap bit in this register is used to indicate if this byte swap is to be disabled for this DMA operation. Word swapping can be disabled in the PCI State register but is normally left enabled.

The ability to reset the Write DMA logic is included to allow current DMA operations to be aborted. Any active PCI operation is immediately terminate gracefully before the FIFO is reset. This bit will is not intended to be used for normal operation of the DMA channel.

9.6.5 DMA Read Length

This is a 16-bit counter used to keep track of the number of bytes to transfer. It is decremented for every byte which is read over the PCI Bus. This register should be initialized after both of the host and local addresses are written, but before the DMA is activated. This register must not be written during an active DMA Read operation.

The DMA is terminated when this counter reaches zero and the data in the FIFO has been written to the local memory. The upper 16-bits of this register are unused and perform no function at this time. The maximum amount of data that can transfer during a single DMA operation is 2^{16} - 1 bytes. If the starting length is zero and the bottom 3-bits of the host address is zero, then the DMA will immediately be considered completed.

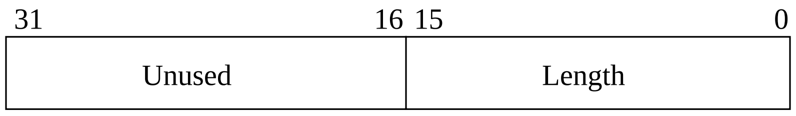

Figure 13. DMA Read Length

9.6.6 DMA Write Length

This is a 16-bit counter used to keep track of the number of bytes to transfer. It is decremented for every byte which is read from the local memory. This register should be initialized after both of the host and local addresses are written, but before the DMA is activated. This register must not be written during an active DMA Write operation.

The DMA is terminated when this counter reaches zero and the data in the FIFO has been written to the host memory. The upper 16-bits of this register are unused and perform no function at this time. The maximum amount of data that can transfer during a single DMA operation is 2^{16} - 1 bytes. If the starting length is zero and the bottom 3-bits of the host address is zero, then the DMA will immediately be considered completed.

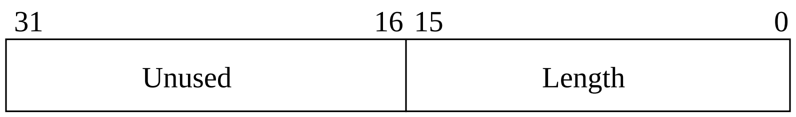

Figure 14. DMA Write Length

9.6.7 DMA Read Local Address

This is the 23-bit register/counter which is used to point to the next location in the adapter's local memory where data is going to be written. All 23-bits are implemented as a counter. The address in this register can be anywhere in first 8 MBytes of the local memory space and does not automatically change any of the producer indexes for either the receive or transmit buffers.

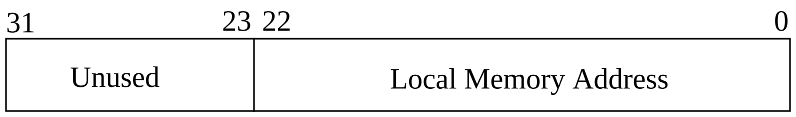

Figure 15. DMA Read Local Address

The initial value of the least significant 3-bits of this register are used along with the least significant 3-bits of the host address register to determine if any byte alignment operations will be necessary.

9.6.8 DMA Write Local Address

This is the 23-bit register/counter which is used to point to the next location in the adapter's local memory where data is going to be read from. All 23-bits are implemented as a counter. The address in this register can be anywhere in first 8 MByte of the local memory space and does not automatically change any of the consumer indexes for either the receive or transmit buffers.

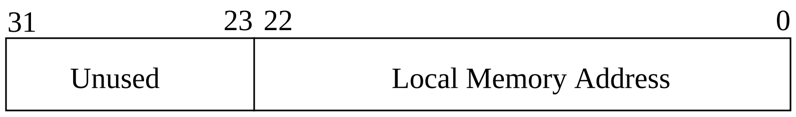

Figure 16. DMA Write Local Address

The initial value of the least significant 3-bits of this register are used along with the least significant 3-bits of the host address register to determine if any byte alignment operations will be necessary.

9.6.9 DMA Read TCP/IP Checksum

The TCP/IP ones-complement checksum is calculated as data is written into the local memory from the internal 64-bit FIFO. This creates a 16-bit checksum result. The checksum registers are managed by the internal processor for each DMA operation.

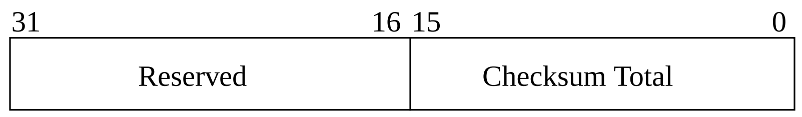

Figure 17. DMA Read TCP/IP Checksum

If the Invert Checksum bit is set in the DMA Read State register, then the 16-bit Checksum Total field will be inverted. This is useful for calculating a checksum for a packet since it must be inverted prior to being inserted into the proper checksum field within the packet.

The Checksum register can be cleared by either writing a zero to the register or by writing a '1' to the bit in the DMA Read Control register which indicates the checksum should be cleared by hardware prior to the start of the DMA. Another bit in the DMA Read Control register is used to determine if the Checksum calculation should be performed during the current DMA operation. This provides the ability for the processor to save the checksum for any temporary DMA activity which consists of data not to be included in the checksum, without having to load and store a copy of the intermediate checksum result.

In situations where the checksum should not start at the beginning of the host buffer space, the software has one of two choices. It can DMA the buffer as one operation, and then subtract the unwanted bytes in software. Or it can create two DMA operations such that the second DMA starts at the point in which the checksum should be calculated.

9.6.10 DMA Write TCP/IP Checksum

The TCP/IP ones-complement checksum is calculated as data is written into the internal 64-bit FIFO from the local memory. This creates a 16-bit checksum result. The checksum registers are managed by the internal processor for each DMA operation.

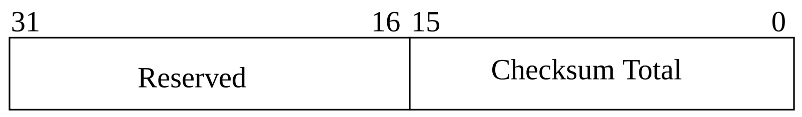

Figure 18. DMA Write TCP/IP Checksum

The Checksum register can be cleared by either writing a zero to the register or by writing a '1' to the bit in the DMA Write Control register which indicates the checksum should be cleared by hardware prior to the start of the DMA. Another bit in the DMA Write Control register is used to determine if the Checksum calculation should be performed during the current DMA operation. This provides the ability for the processor to save the checksum for any temporary DMA activity which consists of data not to be included in the checksum, without having to load and store a copy of the intermediate checksum result.

In situations where the checksum should not start at the beginning of the Ethernet packet, the software has one of two choices. It can DMA the buffer as one operation, and then subtract the unwanted bytes in software. Or it can create two DMA operations such that the second DMA starts at the point in which the checksum should be calculated.

9.7 Local Memory Configuration Registers

The Local Memory Configuration registers are used to manage the usage of the local memory. Some of the registers are set only after initialization, while others are frequently accessed by software and hardware. All of these registers are officially read-write to the processor, but some registers will typically be used in a read-only fashion. The registers which make up the local memory configuration are listed in Table 36.

Table 36. Local Memory Configuration Registers

Offset	Registers	Typical Processor Access	Hardware Access
0C0h-0C3h	Receive Buffer Base	R/W	R/O
0C4h-0C7h	Receive Buffer Producer	R/O	R/W
0C8h-0CBh	Receive Buffer Consumer	R/W	R/O
0CCh-0CFh	Receive Buffer Active Packet Start	R/O	R/O
0D0h-0D3h	Transmit Buffer Base	R/W	R/O
0D4h-0D7h	Transmit Buffer Producer	R/W	R/O
0D8h-0DBh	Transmit Buffer Consumer	R/O	R/W
0DCh-0DFh	Transmit Next Packet Starting Address*	R/W	R/W
0E0h-0E3h	Receive Descriptor Producer	R/O	R/W
0E4h-0E7h	Receive Descriptor Consumer	R/W	R/O
0E8h-0EBh	Receive Descriptor Reference	R/W	R/O
0ECh-0EFh	CPU Priority	R/W	R/O
0F0h-0F3h	Transmit Descriptor Producer	R/W	R/O
0F4h-0F7h	Transmit Descriptor Consumer	R/O	R/W
0F8h-0FBh	Transmit Descriptor Reference	R/W	R/O
0FCh-0FFh	Transmit Next Descriptor Available*	R/O	R/O

* First Implemented in Tigon revision 5.

All of the registers between 0C0h and 0DFh which are used to delineate the receive and transmit portions of the local memory have register layouts as shown in Figure 19.

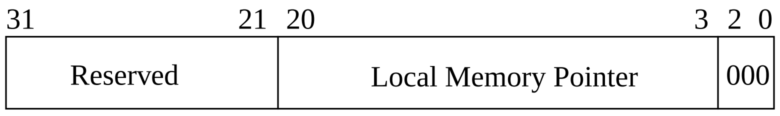

Figure 19. Local Memory Pointer Registers

The 6 registers which are used as Ethernet Descriptor indexes all have register layouts as shown in Figure 20. The Descriptor Base Address is automatically determined by hardware and is provided as read-only during reads of these registers. Thus these registers function as a pointer directly to the address of a specific descriptor.

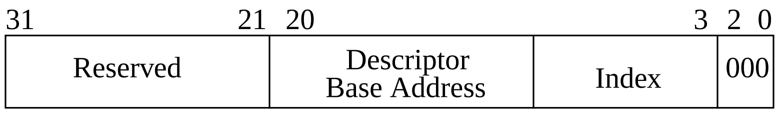

Figure 20. Ethernet Descriptor Index Registers

9.7.1 Receive Buffer Base

The Receive Buffer Base register is a 21-bit register which describes the starting address of the receive buffer. The receive buffer is always described as ending at address 1FFFFFh in the local memory space. For example, a 128k Byte receive buffer would be defined with a value of 1E0000h in this register. Software must initialize this register prior to allowing receive data in from the Ethernet receive interface. The base address can be on any 4 kByte boundary.

! **NOTE:** *If 256k bytes of local SRAM exists instead of 2M byte, the SRAM will be mapped such that it is repeated 8 times in that region. Similarly 1M bytes of local memory will appear repeated twice.*

At any time that any of the hardware read or write DMAs reach the predetermined value of 1FFFFFh, they will be reloaded with the value of the Receive Buffer Base after the next read or write respectively. This helps ensure that all accesses to the receive buffer stay within the boundaries of the receive buffer. Figure 19 shows the layout of this register.

9.7.2 Receive Buffer Producer

The Receive Buffer Producer register is initially set-up by software but afterwards controlled by the hardware. This pointer is generally written after every two 32-bit Ethernet words are received and also at the end of the packet if the packet contains an odd number of words. This register is used in conjunction with the Receive Buffer Consumer register to determine the available space is the receive buffer. Figure 19 shows the layout of this register. This pointer can only point to doubleword address boundaries.

9.7.3 Receive Buffer Consumer

The Receive Buffer Consumer register is entirely managed by software. This pointer is generally updated after a DMA operation completes successfully; thus indicating a portion of the receive buffer can now be freed-up. This register is used in conjunction with the Receive Buffer Producer register to determine the available space is the receive buffer. Figure 19 shows the layout of this register. This pointer can only point to doubleword address boundaries.

9.7.4 Receive Buffer Active Packet Start

This register is only used when the firmware wants to process a packet after the packet start event and prior to the packet complete event. Since the descriptor is not written until the packet completes, this register provides a means to determine where the packet starts in the local memory. To prevent race conditions it is best to read the receive producer register before and after reading this register to make sure that the hardware has not advanced to another packet while this register was read. The number of bytes which are guaranteed to be valid prior to receiving a packet start event is determined in the MAC receive state register.

9.7.5 Transmit Buffer Base

The Transmit Buffer Base register is a 21-bit register which describes the starting address of the transmit buffer. The transmit buffer is always described as ending at the local memory address defined as one word just before the Receive Buffer Base register. For example, a 64k Byte transmit buffer configured behind a 128k Byte receive buffer, would indicate that the Transmit Base register should be written with a value of 0D0000h. Software must initialize this register prior to allowing transmit data in to the Ethernet transmit interface. The base address can be on any 4 kByte boundary.

At any time that any of the hardware read or write DMAs reach Receive Buffer Base register less one word, they will be reloaded with the value of the Transmit Buffer Base after the next read or write respectively. This helps ensure that all accesses to the transmit buffer stay within the boundaries of the transmit buffer. Figure 19 shows the layout of this register.

9.7.6 Transmit Buffer Producer

The Transmit Buffer Producer register is entirely managed by software. This pointer is generally updated after a DMA operation completes successfully; thus indicating more data is now available in the transmit buffer. This register is used in conjunction with the Transmit Buffer Consumer register to determine the available space is the transmit buffer. Figure 19 shows the layout of this register.

9.7.7 Transmit Buffer Consumer

The Transmit Buffer Consumer register is initially set-up by software but afterwards controlled by the hardware. This pointer is generally written after every two 32-bit Ethernet words are transmitted down the Ethernet channel. This register is used in conjunction with the Transmit Buffer Producer register to determine the available space is the transmit buffer. Figure 19 shows the layout of this register.

9.7.8 Transmit Next Packet Starting Address[1]

The Transmit Buffer Next Packet Start register can be optionally used to allow the hardware to determine where the next transmit packet should be placed within the transmit buffer. The firmware should write the length of the packet to this register and then read the result. If the result is zero, then there is not enough room for the packet. Otherwise the result will be the address at which the packet should start. A successful starting address will automatically cause the hardware to reduce the available space by the packet size rounded up to the next 64-bit boundary.

If firmware wishes to check where the next packet would go without having the hardware allocate the space, simply write a length of zero. This would not consume any space in the transmit buffer, but is guaranteed to return the starting address for the next packet.

9.7.9 Receive Descriptor Producer

The Ethernet Receive Descriptor Consumer register is used to indicate to the software which descriptors have been used by the Ethernet receive channel. It is updated by hardware partially before the start of each packet and partially after each packet has been received.

Reads of this register will yield the exact address in the local memory at which the Ethernet descriptor is located for the packet currently being received. Figure 20 shows the layout of this register. The receive descriptor region is located directly below the transmit buffer base address.

[1]First implemented in Tigon revision 5.

9.7.10 Receive Descriptor Consumer

The Ethernet Receive Descriptor Consumer register is used to indicate to the hardware which descriptor is the last one available for the hardware to utilize. It is updated by software at the point at which the received packet descriptor is no longer needed.

Reads of this register will yield the exact address in the local memory at which the Ethernet descriptor of the next packet queued for transfer into host memory can be found. Figure 20 shows the layout of this register. The receive descriptor region is located directly below the transmit buffer base address.

9.7.11 Receive Descriptor Reference

The Ethernet Receive Descriptor Reference register is used by software to control when events are generated. A comparison is made between this register and the Ethernet Receive Descriptor Producer register in order to produce receive start and end events. Software can then chose to get notified by an event after each descriptor is updated by the hardware. See Chapter 4, “Internal Events,” for details on generating these events.

Reads of this register will yield the exact address in the local memory at which the referenced Ethernet descriptor can be found. Figure 20 shows the layout of this register. The receive descriptor region is located directly below the transmit buffer base address.

9.7.12 CPU Priority

This register provides a mechanism for setting the priority of the internal processor when there is arbitration to the local SRAM. The lower 4-bits of this register are R/W and indicate the relative priority of the processor (0=low, 15=high).

In hardware this priority function is implemented by counting the number of clocks the processor has been waiting for an SRAM access. The maximum number of clock cycles in which the CPU will wait before being given access to the SRAM is calculated as 15 - (4-bit CPU Priority register). This priority counter does not start until the CPU has a read or write operation pending. Once a CPU operation completes, then the priority counter starts over.

Instruction fetches and data load/store operations are not distinguished for the purpose of this priority function. All CPU to SRAM accesses share the same CPU priority hardware.

The upper 16-bits of this register provide access to some internal nets for increased testability. These fields are defined in Table 37. Normal firmware can ignore all of these read-only testability fields.

! **NOTE:** *Technically any value between the range 0-15 is allowed, however a value of 15 may force the CPU to cause the DMA channels to starve. It is expected that the Tigon will function best if the priority is left somewhere between 0-7. Only during critical threads of code should the processor bump its priority above 8. The higher the CPU priority, the lower will be the DMA priority*

Table 37. CPU Priority Register

Bits	Field	Description	Access
31-24	Receive Buffer Available	Indicates number of 128 byte memory blocks which are unused in the receive buffer up to a maximum of 255. Prior to Tigon revision 5 maximum was 127.	R/O
23-16	Receive Descriptors Available	Indicates the number of receive descriptors which are unused up to a maximum of 255. Prior to Tigon revision 5 maximum was 127.	R/O
15-14	Flash Region Address Setup*	Number of clock cycles in Flash address region for address to settle before OE or WE is asserted.	R/W
13-10	Flash Region Operation*	Number of clock cycles in Flash address region in which OE or WE are asserted.	R/W
9-8	Flash Region Address Hold*	Number of clock cycles in Flash address region after OE or WE are asserted for bus to tristate.	R/W
7-4	CPU B Priority*	CPU local memory priority (0=low, F=high) Power on default is 0.	R/W
3-0	CPU A Priority	CPU local memory priority (0=low, F=high) Power on default is 0.	R/W

* First implemented in Tigon revision 5.

The upper 16-bits of this register provide access to some internal nets for increased testability. These fields are defined in Table 37. Normal firmware can ignore all of these read-only testability fields.

9.7.13 Transmit Descriptor Producer

The Ethernet Transmit Descriptor Producer register is used to indicate to the hardware that a packet is ready to be send to the Ethernet channel. It is updated by software at the point at which the packet should start transmission. The packet does not have to be completely in the transmit buffer if a flow-through packet model is being utilized.

Reads of this register will yield the exact address in the local memory at which the Ethernet descriptor of the last packet queued for transmission can be found. Figure 20 shows the layout of this register. The transmit descriptor region is located directly below the receive descriptor region.

9.7.14 Transmit Descriptor Consumer

The Ethernet Transmit Descriptor Consumer register is used to indicate to the software which descriptors have been sent to the Ethernet channel. It is updated by hardware after the descriptor has been transmitted.

Reads of this register will yield the exact address in the local memory at which the Ethernet descriptor which is being transmitted can be found. Figure 20 shows the layout of this register. The transmit descriptor region is located directly below the receive descriptor region.

9.7.15 Transmit Descriptor Reference

The Ethernet Transmit Descriptor Reference register is used by software to control when events are generated. A comparison is made between this register and the Ethernet Transmit Descriptor Consumer register in order to produce transmit completed events. Software can then chose to get notified by an event after each descriptor is transmitted or after a group of descriptors have been transmitted. See Chapter 4, "Internal Events," for details on generating these events.

Reads of this register will yield the exact address in the local memory at which the referenced Ethernet descriptor can be found. Figure 20 shows the layout of this register. The transmit descriptor region is located directly below the receive descriptor region.

9.7.16 Transmit Next Descriptor Available[1]

The Ethernet Transmit Next Free Descriptor register is used by software to determine if there is a unused transmit descriptor as well as get a pointer to local memory of that descriptor's location. Hardware determines that there is an unused descriptor by comparing the producer and the consumer. If the producer + 1 is equal to the consumer, then all descriptors are used. Reading this register will either return a null address (i.e. zero) if there are no descriptors available, or it will return the same value as the transmit descriptor producer if there are descriptor available.

[1]First implemented in Tigon revision 5.

9.8 Host FIFO Access

This 12k Byte region is used as a means for the host to write or read directly from the Tigon's DMA FIFOs. It is not expected that this region will be used during more efficient adapter to host models, but it is useful feature when two Tigons want to DMA at high speeds data directly to each other.

! **NOTE:** *Care should be taken not to access these locations during an active Tigon DMA since both processes would compete for the FIFOs and the result would be unpredictable.*

Reads to this region will read the data at the front of the Write DMA FIFO, while writes to this would write data into the Read DMA FIFO. Keep in mind that both of the DMA channels are named after whether they normal Write or Read from the host memory. If a read is performed when no data was present in the FIFO then the value returned will be legal data whose value is not determinable. Similarly is a write is performed into a full FIFO then the data written will be dropped.

A control bit in the PCI State register enables the ability to force PCI retry operations when the Tigon is acting as a PCI target and the FIFO being accessed cannot complete the desired operation. Thus a read from an empty FIFO or a write to a full FIFO would then cause PCI retry operations. The threshold field of each DMA channel is used as the criteria to determine if a retry operation should be issues by the Tigon. To burst data directly into or out of a DMA FIFO, the external source must ensure that it does not source or sink more data than the threshold which was set in the respective DMA channel state register.

9.9 Host DMA Assist Registers

The Host DMA Assist registers are used to control the operation of the DMA Assist logic. This logic is used to read DMA descriptors from local memory in order to keep both of the DMA channels busy.

Table 38. Host DMA Assist Registers

Offset	Registers	Typical Processor Access	Hardware Access
100h-103h	Read Channel High Priority Producer	R/W	R/O
104h-107h	Read Channel High Priority Consumer	R/O	R/W
108h-10Bh	Read Channel High Priority Reference	R/W	R/O
10Ch-10Fh	Read Channel High Next Descriptor Available*	R/O	R/O
110h-113h	Read Channel Low Priority Producer	R/W	R/O
114h-117h	Read Channel Low Priority Consumer	R/O	R/W
118h-11Bh	Read Channel Low Priority Reference	R/W	R/O
11Ch-11Fh	Assist State	R/W	R/O
120h-123h	Write Channel High Priority Producer	R/W	R/O
124h-127h	Write Channel High Priority Consumer	R/O	R/W
128h-12Bh	Write Channel High Priority Reference	R/W	R/O
12Ch-12Fh	Write Channel High Next Descriptor Available*	R/O	R/O
130h-133h	Write Channel Low Priority Producer	R/W	R/O
134h-137h	Write Channel Low Priority Consumer	R/O	R/W
138h-13Bh	Write Channel Low Priority Reference	R/W	R/O
13Ch-13Fh	Reserved		

* First implemented in Tigon revision 5.

The four Assist DMA descriptor rings are fixed in their located in local memory and occupy the 4k byte directly below the bottom of the transmit buffer region. Each ring supports 32 eight-word DMA descriptors or 64 four-word descriptors which totals 1k byte per ring. It is also possible to combine the high and low rings into a single ring which supports double the number of descriptors. The order of the rings is the same order as the DMA Assist registers are defined in Table 38 going from low memory to high memory.

9.9.1 Assist State

The Assist State register is used to control the operation of the DMA assist logic. The assist logic need only be enabled if the DMA descriptors are going to be utilized to control the DMA channels. The firmware always has the option of directly controlling the DMA channels without the use of the assist logic. The fields within this register are shown in Table 39.

Table 39. Assist State Register

Bits	Field	Description	Access
31-16	Reserved	Always 0	
15	Read High Not Empty	At least one DMA descriptor on high priority read DMA descriptor ring has not been completed.	R/O
14	Read Low Not Empty	At least one DMA descriptor on low priority read DMA descriptor ring has not been completed.	R/O
13	Read High Selected	Currently active read DMA is from high priority ring.	R/O
12	Read Busy	Read DMA channel is active.	R/O
11	Write High Not Empty	At least one DMA descriptor on high priority write DMA descriptor ring has not been completed.	R/O
10	Write Low Not Empty	At least one DMA descriptor on low priority write DMA descriptor ring has not been completed.	R/O
9	Write High Selected	Currently active write DMA is from high priority ring.	R/O
8	Write Busy	Write DMA channel is active.	R/O
7-6	Reserved	Always 0	
5	Combine high-low Rings	Combine the high and low rings into a single ring. Use only the high ring register.	R/W
4	Read Mini-Format Descriptor	Use mini DMA descriptor format for Read channel. Prior to Tigon revision 5, both channels must use same descriptor format.	R/W
3	Write Mini-Format Descriptor	Use mini DMA descriptor format for Write channel. Prior to Tigon revision 5, both channels must use same descriptor format.	R/W
2	One DMA Active	Allow only one DMA channel to be active at one time. Alternate channels among equal priorities.	R/W
1	Paused	Stop assist logic from all descriptor processing. Does not affect DMA transfers which are active.	R/W
0	Enable	Enable Assist logic, all assist registers have been initialized.	R/W

9.9.2 Assist Producer

The Assist Producer registers are used as a means for firmware to indicate to the Assist logic which DMA descriptors are ready to be processed by the hardware. Each of the four DMA descriptor rings has its own producer register. Hardware uses this register in conjunction with the consumer and reference registers for the same ring to determine if an event should be generated to the internal processor.

Reads of these registers will yield the exact address in the local memory at which the next free DMA descriptor is located. The upper portion of the address is read-only, thus allowing hardware to manage the roll-over pointer process by truncating the upper bits during writes.

9.9.3 Assist Consumer

The Assist Consumer registers are used as a means for the assist hardware to keep track of which DMA descriptors have been processed. Each of the four DMA descriptor rings has its own consumer register. Hardware also uses this register in conjunction with the producer and reference registers for the same ring to determine if an event should be generated to the internal processor.

Reads of these registers will yield the exact address in the local memory at which the DMA descriptor is located for the active DMA (or next DMA to be started if channel is not active). The upper portion of the address is read-only, thus allowing hardware to manage the roll-over pointer process by truncating the upper bits during writes.

9.9.4 Assist Reference

The Assist Reference registers are used as a means for firmware to indicate to the Assist logic when it wants an event to be generated. Each of the four DMA descriptor rings has its own reference register. Hardware uses this register in conjunction with the producer and consumer registers of the same ring to determine if the indicated DMA descriptor has been completed. An event is generated if the DMA descriptor referenced has completed.

Reads of these registers will yield the exact address in the local memory at which the DMA descriptor is located for the DMA which when completed will generate an event. The upper portion of the address is read-only, thus allowing hardware to manage the roll-over pointer process by truncating the upper bits during writes.

9.9.5 Read Channel High Next Descriptor Available

This register provides hardware support for determining if a read channel high priority descriptor is available. When this register is read, it returns either a zero indicating that no descriptor is available, or an address to the next unused descriptor (which is the same value as read channel high producer register). If the Mini-Descriptor format is being used, then the hardware will always check for the availability of at least 2 descriptors.

9.9.6 Write Channel High Next Descriptor Available

This register provides hardware support for determining if a write channel high priority descriptor is available. When this register is read, it returns either a zero indicating that no descriptor is available, or an address to the next unused descriptor (which is the same value as write channel high producer register). If the Mini-Descriptor format is being used, then the hardware will always check for the availability of at least 2 descriptors.

9.10 CPU Registers

These registers are used to control the operation of the processor internal to the Tigon. The majority of the registers are used during debug operations.

Table 40. CPU Registers

Offset	Registers	Access
140h-143h	CPU A State	R/W
144h-147h	CPU A Program Counter (PC)	R/W
148h-14Bh	CPU A Hardware Breakpoint	R/W
14Ch-14Fh	CPU A Instruction*	R/W
150h-153h	CPU A Adjust Stack*	R/W
154h-157h	CPU A Internal SRAM Indirect Address	R/W
158h-15Bh	CPU A Internal SRAM Indirect Data	R/W
15Ch-15Fh	CPU A Watchdog Clear	W/O
180h-18Fh	CPU A General Purpose Registers 0-31	R/O
240h-243h	CPU B State*	R/W
244h-247h	CPU B Program Counter (PC)*	R/W
248h-24Bh	CPU B Hardware Breakpoint*	R/W
24Ch-24Fh	CPU B Instruction*	R/W
250h-253h	CPU B Adjust Stack*	R/W
254h-257h	CPU B Internal SRAM Indirect Address*	R/W
258h-25Bh	CPU B Internal SRAM Indirect Data*	R/W
25Ch-25Fh	CPU B Watchdog Clear*	W/O
280h-28Fh	CPU B General Purpose Registers 0-31*	R/O

* First implemented in Tigon revision 5.

9.10.1 CPU State

The CPU State register controls a few miscellaneous functions associated with the CPU in addition to indicating if the processor halted. There are several reasons why the CPU may be halted and they are indicated in the upper 2 Bytes of this register.

Table 41. CPU State Register

Bits	Field	Description	Access
31-25	Reserved	Always 0	
24	Bad Memory Alignment	Load or Store instruction was executed with least significant 2 address bits not valid for width of operation. (ex. word load from an odd byte address).	R/O
23	Invalid Instruction Fetch	Program Counter (PC) upper 2-bits indicate Flash or Internal Register access, or bits 29-24 are non-zero.	R/O
22	Invalid Data Fetch	Data reference with bits 29-24 non-zero.	R/O
21	Page 0 Instruction Reference	When enabled, indicates PC is set to lower 128 Bytes of SRAM.	R/O

Bits	Field	Description	Access
20	Page 0 Data Reference	When enabled, indicates data reference to lower 128 Bytes of SRAM.	R/O
19	Invalid Instruction	Invalid instruction fetched.	R/O
18	CPU Halt Instruction Executed	CPU Halt instruction executed.	R/O
17	Hardware Breakpoint	When enabled, indicates hardware breakpoint was reached.	R/O
16	Halt CPU	Set by external host to halt the internal CPU.	R/W
15	Flush Instruction Cache*	Self clearing bit which forces instruction cache to flush.	W/O
14-6	Reserved	Always 0	
5	Enable Watchdog	Enables watchdog reset state machine. Used in conjunction with Watchdog Clear register.	R/W
4	ROM Fail	Asserted on reset and cleared by ROM code after successfully loading code from serial EEPROM or Flash. Afterwards can be used as a software defined bit.	R/W
3	Enable Data Cache	Enable the data cache to operate.	R/W
2	Enable Page 0 Halt	Enable instruction or data references to the first 128 Bytes of SRAM to force the CPU to halt.	R/W
1	Single Step CPU	Advances the CPU's PC one cycle. If halting condition still exists then CPU will again halt, otherwise it will resume normal operation.	R/W
0	Reset CPU	Self clearing bit which resets the internal CPU.	R/W

* First implemented in Tigon revision 5.

9.10.2 CPU Program Counter (PC)

This register can be used to read or write the current Program Counter of the internal CPU. Reads can occur at any time, however writes can only be performed when the CPU is halted. Writes will also clear any pending instruction in the decode stage of the pipeline.

Prior to Tigon revision 5, this register was implemented using bits 31-2. For revision 5 only bits 31-30 and 23-2 are implemented.

9.10.3 CPU Hardware Breakpoint

This register is used to set a hardware breakpoints based on the CPU's PC. If the PC equals this breakpoint register, then the CPU is halted and the appropriate stopping condition is indicated in the CPU State register.

Table 42. CPU Hardware Breapoint

Bits	Field	Description	Access
31-2	Hardware Breakpoint		R/W
1	Reserved	Always 0	
0	Disable Hardware Breakpoint	When high, disables hardware breakpoint. Reset automatically sets this bit.	R/W

9.10.4 CPU Instruction[1]

The CPU Instruction register allows write-only access to the full 32-bit instruction in the decode stage of the CPU pipeline when the CPU is halted. This register is only used by a debugger in order to replace the current instruction (usually a halt instruction) with the original instruction.

9.10.5 CPU Adjust Stack[2]

The CPU Adjust Stack register is used as on offset during data loads or stores to the region just below the internal scratch pad memory. This feature is intended to allow the placement of the stack in the lower address portion of the internal memory. When the stack needs to grow beyond the size allocated in the internal memory (i.e. stack drops below C00000h) the value of this register is subtracted from the original address in order to map the location into the external memory. This allows for the stack to be continued at a 4k Byte boundary in the slower external memory.

Table 43. CPU Adjust Stack

Bits	Field	Description	Access
31-23	Reserved	Always 0	
22-12	Adjust Address Value	Value subtracted from load/store addresses when an access occurs between A00000h and BFFFFFh.	R/W
11-0	Reserved	Always 0	

9.10.6 CPU Internal SRAM Indirect Address

This register is used when the CPU is halted to access the internal scratch pad memory. An address is written to this register followed by a word read or write to the Internal SRAM Indirect Data register.

Table 44. CPU Internal SRAM Indirect Address

Bits	Field	Description	Access
31-11	Reserved	Always 0	
10-2	Internal SRAM Address	Offset into the internal SRAM to be used for the next data operation from a host through the PCI interface.	R/W
1-0	Reserved	Always 0	

9.10.7 CPU Internal SRAM Indirect Data

This register is used when the CPU is halted to access the internal scratch pad memory. An address is written to the Internal SRAM Indirect Address register prior to accessing this register in order to perform a word read or write of the internal scratch pad memory. The entire 32-bit value of this register is considered valid.

[1]First implemented in Tigon revision 5.
[2]First implemented in Tigon revision 5.

9.10.8 CPU Watchdog Clear

This Watchdog Clear register is only used if the Watchdog Reset state machine is enabled in the CPU State register. The watchdog state machine looks for three transitions of the timer bit without having a write occurring to this Watchdog Clear register in order to reset the CPU. The data value written to this register is irrelevant. More detail of the operation of the watchdog timer can be found in Chapter 3.11, “Watchdog Timer.”

9.10.9 CPU General Purpose Registers 0-31

The Internal CPU registers memory region is used as a means for the host to read all of the general purpose registers within the internal processor. These registers are not normally accessible to the host and are included in this region merely to aid in debugging of the internal processor.

The internal processor controls 32 general purpose registers, each of which is 32-bits wide. All of these registers are direct mapped into a 128 byte read-only portion of the 16k byte shared memory region. Since the internal processor is constantly updating its registers, the host can only reliably use this debug region once it has halted the internal processor.

9.11 Ethernet MAC Registers

These registers are used to control the operation of the Ethernet MAC. There are several parameters which are available for performance and compatibility tuning. All of these MAC registers are accessed using the externally provided transmit clock.

Table 45. Ethernet MAC Registers

Offset	Registers	Access
200h-203h	Transmit State	R/W
204h-207h	Transmit Auto-Negotiation	R/W
208h-20Bh	MAC Address High	R/W
20Ch-20Fh	MAC Address Low	R/W
210h-223h	Transmit Random Backoff	R/W
214h-217h	Transmit Lengths	R/W
218h-21Bh	-	
21Ch-21Fh	Thresholds	R/W
220h-223h	Receive State	R/W
224h-227h	Receive Auto-Negotiation	R/W
228h-22Bh	Receive Multicast Filter Hash 1	R/W
22Ch-22Fh	Receive Multicast Filter Hash 2*	R/W
230h-233h	Receive Multicast Filter Hash 3*	R/W
234h-237h	Receive Multicast Filter Hash 4*	R/W
238h-23Bh	SERDES Configuration*	R/W
23Ch-23Fh	PCS Configuration*	R/W
240h-27Fh	-	

* First implemented in Tigon revision 5.

9.11.1 Ethernet Transmit State

The Ethernet Transmit State register is used to control as well as monitor the Ethernet transmit interface of the Tigon. This register can be written at any time to control the operation of the transmit interface. Once initialized, this register is normally only referenced to determine the cause of a Ethernet Transmit Error Event. The upper 8-bits of this register are OR'd together to form the Ethernet Transmit Attention in the Main Event register. Table 46 indicates the different bit positions within this register.

Table 46. Ethernet Transmit State Register

Bits	Field	Description	Access
31-30	Reserved	Always 0	
29	Flow Control XON Received	If flow control attentions are enabled, indicates an XON packet was received. Write '1' to clear.	R/W
28	Flow Control XOFF Received	If flow control attentions are enabled, indicates an XOFF packet was received. Write '1' to clear.	R/W
27	FIFO Write Overrun	Fatal error - too many words written into FIFO.	R/O
26	FIFO Write Underrun	Fatal error - too few words written into FIFO.	R/O
25	FIFO Read Overrun	Fatal error - too many words read from FIFO.	R/O
24	FIFO Read Underrun	Fatal error - no words available in FIFO.	R/O
23-21	Reserved		
20	Transmit Currently XOFF'd	Transmit traffic stopped due to flow control.	R/O
19-14	Reserved		
13	Flow Control Long Duration	Send XOFF of FFFFh vs. 01FFh duration.	R/W
12	Carrier	Carrier currently active on the media.	R/O
11	Disable Buffer Compare	Disable transmit check for entire packet ready before starting packet transmission.	R/W
10	Enable Big Backoffs	Enable half-duplex backoff of twice the normal range after 2 collisions.	R/W
9	Enable Flow Control Attns	Enable Receive State attentions indicating a flow-control packet was sent.	R/W
8	Enable Sending Flow Control	Enable transmission of flow-control packets in hardware.	R/W
7	Enable 8/10 Encoding	Enable 1000Base 8/10 encoding and decoding.	R/W
6	Enable 1000Base	Enable 1000Base MAC operations (vs. MII).	R/W
5	Enable Full Duplex	Enable full-duplex operation of the MAC.	R/W
4	Enable GMII	Enable Gigabit MII interface.	R/W
3	Enable Check Carrier	Enable Carrier to be checked just prior to packet start.	R/W
2	Stop Transmit after next Packet	Disable transmit MAC gracefully at next inter-packet gap.	R/W
1	Enable Transmit	Enable transmit MAC operation.	R/W
0	Reset Ethernet MAC	Reset entire MAC. This bit is self-clearing.	W/O

9.11.2 Ethernet Transmit Auto-Negotiation

This register is initialized by software and determines what data value should be sent during the l000Base auto-negotiation process. A similar register is available which indicates the value of the received auto-negotiation data.

Table 47. Ethernet Transmit Auto-Negotiation

Bits	Field	Description	Access
31-16	Reserved	Always 0	
15-0	Transmit Auto-Negotiation Data	2 Byte field which is sent during auto-negotiation. Most significant byte is sent first.	R/W

9.11.3 Ethernet MAC Address

The Ethernet MAC needs to be initialized with the 6-byte MAC address in order to perform hardware receive packet filtering. When operating the receive MAC in promiscuous mode, no receive filtering is performed. This address is also used as the source address for sending flow-control packets.

Table 48. Ethernet MAC Address High

Bits	Field	Description	Access
31-16	Reserved	Always 0	
15-0	MAC Address High	Upper 2-bytes of this node’s MAC address. For Alteon, this field is currently 0060h.	R/W

Table 49. Ethernet MAC Address Low

Bits	Field	Description	Access
31-0	MAC Address Low	Lower 4-byte of this node’s MAC address. For Alteon, this field is currently CFxxxxxxh, where ‘x’ represents the lower 3-bytes of the serial number.	R/W

9.11.4 Ethernet Transmit Random Backoff

This register is used to initialize the random backoff interval generator. It is implemented as a 10-bit linear feedback shift register as follows:

random[9:0] = (random <<1) ^ ((random[9]) ? 321h : 0);

If the random generator is initialized to zero, then it will always remain a zero indicating that a no backoff internal is always selected. It is recommended that this field be initialized with the same value that is written to the MAC Address Low register in order to create additional randomness to the initial seed.

Table 50. Ethernet Transmit Random Backoff

Bits	Field	Description	Access
31-10	Reserved	Always 0	
9-0	Random Backoff Seed	For half-duplex, initialize with any non-zero seed.	R/W

9.11.5 Ethernet Transmit Lengths

This register sets up a number of fields which control the operation of the MAC. This register is typically initialized to 251Fh when in full-duplex mode and 241Fh when in half-duplex mode.

Table 51. Ethernet Transmit Lengths

Bits	Field	Description	Access
31-16	TAG Type	This field is used during the optional transmit 802.1Q TAG insertion and replacement.	R/W
15	Reserved	Always 0	
14-13	InterPacket Gap CRS Length	This field when incremented and then multiplied by 2 indicates the number of bytes from the end of the InterPacket Gap (IPG) at which the incoming Carrier is ignored. It is recommended to ignore Carrier during the last 1/3 of the IPG interval.	R/W
12-8	InterPacket Gap Length	This field when incremented and then multiplied by 2 indicates the number of bytes of the entire IPG. In half-duplex or 1000Base mode, an additional 2 bytes of IPG will be inserted beyond the value of this field.	R/W
7-0	Slot Time Length	This field when incremented and then multiplied by 2 indicates the number of bytes of the slot time. Use FFh for 1000 half-duplex and 1Fh for all other modes.	R/W

9.11.6 Ethernet Thresholds

This register defines a number of thresholds which control both notification of low resources as well as when to send flow-control XON or XOFF messages. All of these fields when used by the hardware have three bits of 111b appended to the right end of the threshold value such as to modify the threshold from a 5-bit field to a 8-bit field with a minimum value of 15 and a maximum value of 255. If any field is set with all bits to zero, then that condition will always remain false. Prior to Tigon revision 5, only 2-bits where appended instead of 3-bits reducing the maximum value to 127. There was also no disable condition, thus making the minimum value 3. This register has no power-on default and must be initialized by firmware.

Table 52. Ethernet Thresholds

Bits	Field	Description	Access
31-30	Reserved	Always 0	
29-25	Receive Buffer Attention	When the number of unused 128 Byte receive buffers blocks drops below this threshold, an optional Receive State register buffer attention is generated.	R/W
24-20	Receive Descriptor Attention	When the number of unused receive descriptors drops below this threshold, an optional Receive State register descriptor attention is generated.	R/W
19-15	Receive Buffer XOFF	When the number of unused 128 Byte receive buffers drops below this threshold, an optional XOFF is sent.	R/W
14-10	Receive Descriptor XOFF	When the number of unused receive descriptors drops below this threshold, an optional XOFF is sent.	R/W
9-5	Receive Buffer XON	When the number of unused 128 Byte receive buffers is >= this threshold, an optional XON is sent.	R/W
4-0	Receive Descriptor XON	When the number of unused receive descriptors is >= this threshold, an optional XON is sent.	R/W

9.11.7 Ethernet Receive State

The Ethernet Receive State register is used to control as well as monitor the Ethernet receive interface of the Tigon. This register can be written at any time to control the operation of the receive interface. Once initialized, this register is normally only referenced to determine the cause of a Ethernet Receive Error Event. The upper 8-bits of this register are OR'd together to form the Ethernet Receive Attention in the Main Event register.Table 53 indicates the different bit positions within this register.

Table 53. Ethernet Receive State Register

Bits	Field	Description	Access
31	Receive Descriptor Attention	Latched condition only if descriptor attentions enabled	R/W
30	Receive Buffer Attention	Latched condition only if buffer low attentions enabled.	R/W
29	Flow Control XON Sent	Latched condition only if flow control attentions are enabled.	R/W
28	Flow Control XOFF Sent	Latched condition only if flow control attentions are enabled.	R/W
27	FIFO Overrun	Fatal FIFO error which requires a reset to clear.	R/O
26	Auto-Negotiation Changed	Latched attention is generated if auto-negotiation data value of commands have changed.	R/W
25	Link State Error	Latched attention if excessive decoding errors.	R/W
24	Link Ready Changed	Latched attention if Link Ready input changed.	R/W
23	Reserved		
22	Enable Low Resource Attns*	Enable attentions when below threshold for buffers or descriptors.	R/W
21	Enable Flow Control Attns	Enable Transmit State attentions indicating a flow-control packet was received.	R/W
20	Remote Transmitter XOFF'd	A previously sent XOFF timer has not expired yet.	R/O
19	Receiving Link-Configuration	Currently receiving link-configuration commands.	R/O
18	Current Link Ready	Currently have valid link.	R/O
17	Currently Blocking Reception	Currently blocking reception due to low resources.	R/O
16	Enable Receiving Flow Control	Enable reception of flow-control packets in hardware.	R/W
15	Link Ready Polarity High†	When set, assume Link Ready input pin is active high.	R/W
14-8	Keep packet Length	Indicates the number of double-words into a packet before the packet will be kept, a receive descriptor will be used, and software has the option of early packet notification. Shorter packets will be discarded without having used a descriptor unless they achieve at least 64 Bytes in length. Normal value is 8.	R/W
7-4	Maximum Jams	In half duplex mode, indicates the number of JAMs the transmit logic is allowed to generate due to the receive resources inability to handle the inbound packet.	R/W
3	Promiscuous Mode	Accept all inbound packets, resource permitting.	R/W
2	Stop Receive after next Packet	Disable receive gracefully at next inter-packet gap.	R/W
1	Enable Receive	Enable receive MAC operation	R/W
0	Reserved		

* Last implemented in Tigon revision 4.

† First implemented in Tigon revision 5.

9.11.8 Ethernet Receive Auto-Negotiation

This register is used by software and determines what data value is being received during the l000Base auto-negotiation process. A similar register is available which indicates the value of the transmit auto-negotiation data.

Table 54. Ethernet Receive Auto-Negotiation

Bits	Field	Description	Access
31-16	Reserved	Always 0	
15-0	Received Auto-Negotiation Data	2 Byte field which is received during last valid auto-negotiation. Most significant byte is received first.	R/O

To ensure stability in the received auto-negotiation, software should clear any prior receive attention prior to reading this register three consecutive times. Afterwards there should be no decoding error or Link State Changed attentions and there should be a auto-negotiation attention pending (due to the continuous reception of auto-negotiation commands).

9.11.9 Ethernet Multicast Filter

The Multicast filter registers are used to help discard unwanted multicast packets as they are received from the external media. The destination address is fed into the normal CRC algorithm in order to generate a hash function. The most significant bits of the CRC are then used without any inversion in reverse order to index into a hash table which is comprised of these Multicast Filter registers. If the CRC is calculated by shifting right, then the rightmost bits of the CRC can be directly used with no additional inversion or bit swapping required. See Chapter 6.7, “Ethernet CRC Calculation” for more details on the CRC algorithm.

Prior to Tigon revision 5, only Multicast Filter 1 register was implemented provided a hash table with 32 entries. Thus only 5-bits are used from the CRC.

Tigon revision 5, uses all four Multicast Filter register such that register 1 bit-32 is the most significant hash table entry and register 4 bit-0 is the least significant hash table entry. This follows the normal big-endian ordering used throughout the Tigon. Since there are 128 hash table entries, 7-bits are used from the CRC.

The Multicast Filter registers are ignored if the receive MAC is in promiscuous mode.

9.11.10Ethernet SERDES Configuration[1]

This register directly controls the inputs to the gigabit serializer/deserializer (SERDES) provided by LSI logic. Most of these bits are used only for testing purposes and their usage is defined in the GigaBlaze manual provided by LSI. The register is set to 0x01DB after a reset.

CAUTION: *This register definition is still be subject to change.*

Table 55. Ethernet SERDES Configuration

Bits	Field	Description	Access
31	rxlockr	Receive clock generator lock.	R/O
30	txlockr	Transmit clock generator lock.	R/O
29	passn	Self-test pass active low output.	R/O

[1]First implemented in Tigon revision 5.

Bits	Field	Description	Access
28-26	Reserved	Always 0	
25-24	tp	Transmitter clock generator pole adjust.	R/W
23-22	tz	Transmitter clock generator zero adjust.	R/W
21-20	rp	Receiver clock generator pole adjust.	R/W
19-18	rz	Receiver clock generator zero adjust.	R/W
17-16	emph	Output driver emphasis.	R/W
15-13	selftest	Select self-test mode.	R/W
12	scantest	Scan test enable.	R/W
11	pdownt	Power down transmitter.	R/W
10	pdownr	Power down receiver.	R/W
9	pdownb	Power down bias generator.	R/W
8	rdws	Receive data bus width is 20-bits.	R/W
7	tdws	Transmit data bus width is 20-bits.	R/W
6	ensyncdet	Enable receive sync. detect.	R/W
5	syncpol	Enable receive sync. on both K28.5 polarities.	R/W
4	wrapbackn	Wrapback active low.	R/W
3	loopbackn	Loopback active low.	R/W
2	lockrefn	Lock to reference clock.	R/W
1	Enable Internal PHY	Enables internal gigabit SERDES.	R/W
0	reset	Reset gigabit SERDES.	R/W

9.11.11Ethernet PCS Configuration[1]

This register is used only during testing of the physical components of the Ethernet interface. The output of the PCS stage will be a fixed 20-bit pattern which can be sent repeatedly when in this test mode. During normal operation, this register would be initialized to zero.

Table 56. Ethernet PCS Configuration

Bits	Field	Description	Access
31-21	Reserved	Always 0	
20	Enable PHY Test Mode	When asserted sends 20-bit data patterns repeatedly.	R/W
19-0	PHY Test Data Pattern	20-bit pattern used during PHY testing.	R/W

[1]First implemented in Tigon revision 5.

9.12 General Communications Region

The registers described in this section are used in communication between the host and the processor internal to the Tigon chip. All of the communication registers are available to be defined by software protocols.   Table 57 lists all the communication registers along with the address offset into the shared-memory region at which the register is located.

Table 57. General Communications Region

Host Offset	Local Offset	Register	Access
400h-47Fh	000h-07Fh	CPU Page 0	R/W
480h-48Fh	080h-08Fh	Ethernet MAC Statistics	R/W
500h-507h	100h-107h	Mailbox 0	R/W
508h-50Fh	108h-10Fh	Mailbox 1	R/W
510h-5EFh	110h-1EFh	...	R/W
5F0h-5F7h	1F0h-1F7h	Mailbox 30	R/W
5F8h-5FFh	1F8h-1FFh	Mailbox 31	R/W
600h-7FFh	200h-3FFh	128 Software Defined Words	R/W

None of these registers are actually located inside the Tigon to save on the size of the ASIC. Accesses to these registers from the host will cause an access to a reserved 1k byte region of the local memory. Accesses to these registers from the internal processor is done by directly addressing this 1k Byte region starting at address 000000h in the local memory. All accesses to all registers in this region must be word accesses.

9.12.1 CPU Page 0

This 128 Byte region is similar to the software defined region in that it is accessible to the host as well as the Tigon's processor. It can be used for any purpose. However, for enhanced error checking, the Tigon is normally set up to halt the internal processor if an instruction or data fetch is performed to this region. If this region is to be used, it is important to make sure the internal processor will not be halted when it accesses this region. See the CPU State register for more information on enabled/disabling halts during accesses to this region.

9.12.2 Ethernet MAC Statistics

There are several statistics which are managed by the Ethernet MAC hardware. These statistics are stored in this 128 Byte region. Software must initialize this region to zero prior to enabling the Ethernet interface.

Table 58. MAC Statistics

Host Offset	Local Offset	Statistic
480h-483h	080h-083h	TX Excess Collisions
484h-487h	084h-087h	TX Collision 1 - Start of Collision Histogram
488h-48Bh	088h-08Bh	TX Collision 2
48Ch-48Fh	08Ch-08Fh	TX Collision 3
490h-493h	090h-093h	TX Collision 4
494h-497h	094h-097h	TX Collision 5
498h-49Bh	098h-09Bh	TX Collision 6
49Ch-49Fh	09Ch-09Fh	TX Collision 7
4A0h-4A3h	0A0h-0A3h	TX Collision 8
4A4h-4A7h	0A4h-0A7h	TX Collision 9
4A8h-4ABh	0A8h-0ABh	TX Collision 10
4ACh-4AFh	0ACh-0AFh	TX Collision 11
4B0h-4B3h	0B0h-0B3h	TX Collision 12
4B4h-4B7h	0B4h-0B7h	TX Collision 13
4B8h-4BBh	0B8h-0BBh	TX Collision 14
4BCh-4BFh	0BCh-0BFh	TX Collision 15 - End of Collision Histogram
4C0h-4C3h	0C0h-0C3h	TX Late Collisions
4C4h-4C7h	0C4h-0C7h	TX Defer Packet Start*
4C8h-4CBh	0C8h-0CBh	TX CRC Errors (when CRC not appended)*
4CCh-4CFh	0CCh-0CFh	TX Underrun
4D0h-4D3h	0D0h-0D3h	TX Carrier Not Asserted
4E0h-4E3h	0E0h-0E3h	RX Dropped Packet - Unicast Filter*
4E4h-4E7h	0E4h-0E7h	RX Dropped Packet - Multicast Filter*
4E8h-4EBh	0E8h-0EBh	RX Processed Flow Control Packet*
4ECh-4EFh	0ECh-0EFh	RX Dropped Packet - Lack of Resources*
4F0h-4F3h	0F0h-0F3h	Reserved
4F4h-4F7h	0F4h-0F7h	RX Kept Packet - Broadcast*
4F8h-4FBh	0F8h-0FBh	RX Kept Packet - Multicast*
4FCh-4FFh	0FCh-0FFh	RX Kept Packet - Unicast*

* Not functional in until Tigon revision 5.

9.12.3 Mailbox 0-31

These mailbox registers function much like the software defined communication registers except that they have some hardware associated with monitoring them in order to detect when they've been written to by the host. Each mailbox is a 64-bit field. When the host writes to the least significant word of one of these mailboxes, a bit gets set in the Mailbox Event register corresponding to the mailbox number which was written. The most significant word of the mailbox can be written without causing a mailbox event to be generated. Since there are 32 mailboxes, the Mailbox Event register is 32-bits in size.

Mailbox 0 has another special meaning since any host write to that mailbox will also cause any PCI interrupt signal to get cleared. This provides a mechanism which requires the least number of host operations to pass a message to the internal processor as well as to turn off the interrupt.

It is expected that the various host buffer descriptor indexes which are written to by the host would be located in these mailboxes. Indexes which are read by the host can be located in the mailbox region, but it would be more consistent with the intent of the Tigon to locate them in the Software Defined portion of the communication registers.

9.12.4 Software Defined Words

This region provides 512Bytes which can be defined and utilized in any fashion by software. These locations would be well suited to information which is read by the host either in a polling fashion or after an interrupt. Software should make sure that no word in this region is writable by both the host and the internal processor unless semaphores are somehow implemented. Hardware does not enforce this policy, therefore it is up to software to adopt this convention.

9.13 Local Memory Window

The 2k Byte region is used by the host to access any portion of the adapter's local memory. A Window Base Address register is located in the General Control portion of the shared-memory to determine which portion of the local memory is visible to the host. This Window is movable at any time by the host.

It is expected that this window will be used to download the internal processor's run-time code, as well as for general adapter debugging. The internal processor is capable of retrieving code from the Serial EEPROM or Flash on the adapter, but this is expected to only contain diagnostic functions. By requiring the host to download the run-time code, it provide a natural means to keep the host driver and the adapter software synchronized. This is true since at any time an enhancement occurs in the communication between the host and the adapter, both software portions would change.

10 Signal Descriptions

10.1 PCI Interface

The following PCI signals are implemented by the Tigon. All signals are based on the PCI clock. Additional information regarding each signal can be obtained from the PCI specification.

Table 59. PCI Signals

Signal	Description
PClk	Clock input provides timing for all transactions on PCI bus. Tigon revision 5 supports frequencies between 0-66 MHz.
P66en	Indicates PCI bus is operating in the 33-66 MHz range rather than 0-33 MHz range.
PReset#	Asynchronous reset input from PCI bus.
PAD[63:0]	Multiplexed address and data bus.
PCBE[7:0]	Multiplexed command and byte enables.
PPar	Even parity covering PAD[31:0] and PCBE[3:0].
PPar64	Even parity covering PAD[63:32] and PCBE[7:4].
PFrame#	Driven by current master at beginning of transaction.
PIRdy#	Initiator Ready indicates that current master is ready to transfer data word.
PTRdy#	Target Ready indicates that current target is ready to transfer a data word.
PStop#	Stop indicates current target is requesting master to end transaction.
PIDsel	Initialization Device Select is used as a chip select during configuration cycles.
PDevsel#	Device Select indicates target device has detected its address as the target to the current transaction.
PReq#	Request indicates that this chip desires to be a bus master.
PGnt#	Grant indicates to this chip that it has been granted as the bus master.
PReq64#	Master is requesting a 64-bit PCI bus transaction.
PAck64#	Target is acknowledging a 64-bit PCI bus transaction.
PPerr#	Indicates a parity error on a data word was detected.
PSerr#	Indicates a parity error on an address was detected.
PIntA#	Open drain interrupt signal to system.

10.2 Local Memory Interface

The following local memory signals are implemented by the Tigon. All signals are based on the Local Clock.

Table 60. Local Memory Signals

Signal	Description
LClk	Local Clock input provides timing for all local memory operations. Tigon supports a maximum frequency of 50 MHz. Tigon revision 5 uses an internal clock doubler and can be placed in a mode in which it bypasses this doubler and accepts a 100 MHz clock directly. The input duty cycle should be equal to or better than 60/40.
LClk_1X	Tie this input high if LClk input is 100 MHz, otherwise a low indicates LClk is 50 MHz. This input bypasses the internal clock doubler and has a very light internal pull-down.
PLLVSS	Filtered ground for LClk PLL.
PLLVDD	Filtered 3.3 volt power for LClk PLL.
PLLen	Assert this input high to enable the PLL after power is stable. See LSI’s application notes for greater detail.
LAddr[22:3]	Provides address to local memory.
LDat[63:0]	Local memory bi-directional data bus with very light internal pull-downs.
Loe#[1] / Lsoe#	Output enable for odd SRAM banks as well as Flash device. If synchronous memory is used, then this signal becomes the output enable for all devices including those in the flash region.
Loe#[0] / Lsce#	Output enable to even SRAM banks. If synchronous memory is used, then this signal becomes the global chip enable for all banks synchronous SRAM.
Lwe#[1]	Write enable to odd SRAM banks as well as Flash device.
Lwe#[0] / LsADSC#	Write enable to even SRAM banks. If synchronous memory is used, then this signal becomes the global address strobe (adsc) for all banks synchronous SRAM.
Lce#[7:0] / Lsbw#[7:0]	Chip enables for all local SRAM banks. Lce#[7] is for LDat[63:56], while Lce#[0] is for LDat[7:0]. If synchronous memory is used, then these signals becomes the byte write enables for all banks synchronous SRAM.
Lce#[8]	Chip enable for Flash memory.
LED_Traffic	Software controlled 12mA output which can be used to drive an LED. Output is low after reset.
LED_Link	Software controlled 12mA output which can be used to drive an LED. Output is low after reset.
UART_in	Asynchronous UART input with start bit active low and stop bit active high.
UART_out	Asynchronous UART output with start bit active low and stop bit active high.
EE_Clk	Software controlled serial EEPROM clock output. Output is low after reset.
EE_Dat	Software controlled serial EEPROM bi-directional data pin which has a very light internal pull-down. Pin is tri-stated after reset.
Misc[2]	Software controlled 12mA output which can be used for any board level purpose. Prior to Tigon revision 5, this output had been used to indicate loopback to the external SERDES component. Output is low after reset.
Misc[1:0]	Software controlled 4mA bi-directional signals which can be used for any board level purpose. Pins are tri-stated after reset.
JTCK	JTAG test clock input which requires an external pull-down resistor.
JTRST	JTAG test reset input with a very light internal pull-down. If external connections are made, then this pin will need an external pull-down resistor.

Table 60. Local Memory Signals

Signal	Description
JTMS	JTAG test mode select with a very light internal pull-up. If external connections are made, then this pin will need an external pull-up resistor.
JTDI	JTAG test serial data input with a very light internal pull-up. If external connections are made, then this pin will need an external pull-up resistor.
JTDO	JTAG test serial data output.
PROCMON	This output is used during testing to check the process variation.
IDDTN	This signal when high is used to disable all internal pull-ups or pull-downs. Externally requires a pull-down resistor.
TN	This signal when low signal is used to tristate all outputs. This pin has a very light internal pull-up.

10.3 Ethernet Interface

The following signals comprise of the Ethernet transmit and receive interface. Many of the signals have dual use depending on whether the Tigon is using the internal SERDES, external SERDES, GMII, or MII.

Table 61. Ethernet Signals

Signal	Description
TClk_in	Transmit clock input used for internal SERDES, external SERDES, and GMII modes. Depending on the mode, it will either be 62.5 MHz or 125 MHz. This pin must have a valid clock signal during reset.
TClk_out	When TClk_Select indicates GMII or External SERDES, this signal contains an inverted version of TClk_in, but it is aligned with the data to provide proper setup and hold time. In all other modes this signal can contain a divided by 4 version of the internal local clock. This allows the possibility to use this output for the MII clock. This output can be independently tri-stated if it is not used.
RClk0 / MII_TClk	In external SERDES mode this input is used as the second receive input clock. In MII mode this input is used as the transmit clock.
RClk1	Receive clock input used for internal SERDES, external SERDES, and GMII modes. Depending on the mode, it will either be 62.5 MHz or 125 MHz. This pin must have a valid clock signal during reset.
MII_Clk	MII or GMII management interface clock output. Output is low after reset.
MII_Dat	MII or GMII managmenet interface bi-directional data pin which has a very light internal pull-down. Pin is tri-stated after reset.
LnkRdy	In internal SERDES, external SERDES or GMII modes this input should be used to indicate a valid signal is being received. If the PHY produces a signal detect (SD) this would be a good choice for this input.
COL	In GMII or MII modes this input is used as to indicate collision.
CRS	In GMII or MII modes this input is used as to indicate carrier sense.
TDat[9] / Tx_Error	In external SERDES mode this output is data bit 9 (first bit transmitted). In MII or GMII mode, this output represents transmit error.
TDat[8] / Tx_Enable	In external SERDES mode this output is data bit 8. In MII or GMII mode, this output represents transmit enable.
TDat[7:4]	In external SERDES or GMII modes these outputs are data bits 7-4.
TDat[3:0]	In external SERDES, GMII, or MII modes these outputs are data bits 3-0.

Signal	Description
RDat[9] / Rx_Error	In external SERDES mode this input is data bit 9 (first bit received). In MII or GMII mode, this input represents receive error.
RDat[8] / Rx_DatVal	In external SERDES mode this input is data bit 8. In MII or GMII mode, this input represents receive data valid.
RDat[7:5]	In external SERDES or GMII modes these inputs are data bits 7-5.
RDat[4] / MII_Link	In external SERDES or GMII modes this input is data bit 4. In MII mode this signal is used as the link input.
RDat[3:0]	In external SERDES, GMII, or MII modes these inputs are data bits 3-0.
	The following are described in LSI's GigaBlaze™ G10™ documention:
txp / txn	Differential output pair for internal SERDES. Must be externally terminated within the range of 50 - 75 ohms.
rxp / rxn	Differential input pair for internal SERDES. Must be externally terminated within the range of 50 - 75 ohms.
bgvss, gdvss, rxbvss, rxvss, txbvss, txvss	Analog ground for the internal SERDES.
bgvdd, rxbvdd, rxvdd, txbvdd, txvdd	Individually filtered analog 3.3 volt power for the internal SERDES.
iref	Internal SERDES bias generator input current reference.
termres	Connect to a resistor with a value 4x of the txp/txn termination value. The other end of the resistor goes to the same place at which bgvdd is connected.
vtxlo	Internal SERDES minimum output voltage swing comprised of a voltage divider between the analog power and ground.

NOTE: *The bit-numbering convention is different for SERDES components. The transmit and receive data paths may need to be bit-reversed when connected to these components. The Tigon assumes bit-9 is the first bit transmitted and received when in external SERDES mode.*

Table 62 shows the usage of the pins in the four different modes supported. The board design can support more than one mode if the sets of pins have non-overlapping functions (such as internal SERDES and MII). Overlapping tends to occur with the LnkRdy pin if both an internally and externally controlled PHY exists. It would be up to the board level design to multiplex this pin as necessary.

If the TClk_in pin is connected to a 125 MHz clock source, the internal SERDES can still be used since there is an automatic internal TClk_in/2 when the control signal TClk_Select indicates External SERDES or GMII.

Table 62. PHY Dependent Pin Usage

Pin Name	Internal SERDES	External SERDES	MII	GMII
TClk_in	62.5 MHz / 125 MHz	TBC (125 MHz)	any clock	125 MHz
TClk_out	25.0*	~TBC (125 MHz)	25.0*	GTX_CLK (125 MHz)
RClk0 / MII_TClk	any valid signal	RBC0 (62.5 MHz)	TX_CLK (2.5 or 25 MHz)	any valid signal
RClk1	any clock	RBC1 (62.5 MHz)	RX_CLK (2.5 or 25 MHz)	RX_CLK (125 MHz)
LnkRdy	Signal Detect	Signal Detect	-	Link
COL	-	-	COL	COL
CRS	-	-	CRS	CRS
TDat[9] / Tx_Error	-	TXDATA[0]†	TX_ER	TX_ER
TDat[8] / Tx_Enable	-	TXDATA[1]†	TX_EN	TX_EN
TDat[7:4]	-	TXDATA[2:5]†	-	TXD[7:4]
TDat[3:0]	-	TXDATA[6:9]†	TXD[3:0]	TXD[3:0]
RDat[9] / Rx_Error	-	RXDATA[0]†	RX_ER	RX_ER
RDat[8] / Rx_DatValid	-	RXDATA[1]†	RX_DV	RX_DV
RDat[7:5]	-	RXDATA[2:4]†	-	RXD[7:5]
RDat[4] / MII_Link	-	RXDATA[5]†	Link	RXD[4]
RDat[3:0]	-	RXDATA[6:9]†	RXD[3:0]	RXD[3:0]

* Actual internal LClk/4. May be used as source for MII device if internal LClk is 100 MHz.

† Notice bit reversal since SERDES components use bit-0 as the most significant bit.

11 Timing

11.1 PCI Interface

The timing for the PCI interface was derived from the PCI specification. Below is a abbreviated list of those timing parameters to help show the basic requirements of the Tigon. The PCI specification should be consulted to determine the test conditions at which these results should be measured.

Table 63. PCI Timing

Parameter	Min	Max	Test Load
Output delay from PClk	2.41 ns	5.90 ns	50 pf
Input setup to PClk	2.43 ns	-	-
Input hold to PClk	-0.14 ns	-	-

11.2 Local Memory Interface

The timing for the local memory interface is more complex than for any of the other interfaces. These signals are carefully controlled in order to allow most standard 12 ns asynchronous SRAM to be used for local memory. The timing for the signals is entirely based on the rising or falling edge of LClk as indicated in Table 64. All timing in this table correspond to worst case timing for the Tigon. If external synchronous SRAM is used, then a +- 0.25ns should be added to account for the clock skew of the PLL internal to the Tigon.

Table 64. Local Memory Timing

Parameter	LClk Edge	Min	Max	Test Load
LAddr output	rising	1.22 ns	4.65 ns	90 pf
Lce# output	rising	1.72 ns	4.29 ns	30 pf
Lwe#[1] / Lwe#[0] H -> L output	falling	2.60/2.70 ns	4.45/4.81 ns	90 pf
Lwe#[1] / Lwe#[0] L -> H output	falling	2.17/2.30 ns	3.50/3.88 ns	90 pf
Loe#[1] / Loe#[0] H -> L output	falling	2.75/2.76ns	4.72/4.71 ns	90 pf
Loe#[1] / Loe#[0] L -> H output	rising	3.01/2.48 ns	4.36/4.05 ns	90 pf
Lsoe# H -> L output	rising	2.85 ns	5.25 ns	90 pf
Lsoe# L -> H output	rising	2.71 ns	4.43 ns	90 pf
LDat L,H -> L,H output	rising	2.23 ns	6.05 ns	30 pf
LDat Z -> L,H output	falling	3.29 ns	6.29 ns	30 pf
LDat L,H -> Z output	rising	3.09 ns	5.19 ns	30 pf
LDat input setup	rising	1.45 ns	1.89 ns	N/A
LDat input hold	rising	-0.41 ns	0.00 ns	N/A

11.3 Transmit Ethernet Interface

All of the non-clock outputs in the Ethernet transmit interface have the same timing and support a maximum DC current of 4mA.

Table 65. Transmit Ethernet Timing

Parameter	Min	Max	Test Load
Output delay from TClk_i	2.22 ns	4.62 ns	20 pf
Output delay to TClk_o	2.41 ns	4.51 ns	20 pf

11.4 Receive Ethernet Interface

All of the inputs in the Ethernet receive interface have the same timing.

Table 66. Receive Ethernet Timing

Parameter	Min	Max
RDat input setup	-0.06 ns	-0.07 ns
RDat input hold	0.32 ns	0.50 ns

12 Package and Pinout

12.1 Package

Details on the Tigon package can be found in LSI’s documentation under the package code II51. This is a 388 pin PBGA with fixed power and ground pins connected to internal planes.

12.2 Pinout

Pin#	Pin Name	Type
AC14	PClk	Input PCI
AD9	P66en	Input PCI
T1	PReset#	Input PCI
AD5	PPar	Bidi PCI
AD16	PPar64	Bidi PCI
AF3	PFrame#	Bidi PCI
AD2	PIRdy#	Bidi PCI
AD4	PTRdy#	Bidi PCI
AE4	PStop#	Bidi PCI
Y4	PIDsel	Input PCI
AF4	PDevsel#	Bidi PCI
U1	PReq#	Output PCI
U3	PGnt#	Input PCI
AC12	PReq64#	Bidi PCI
AE12	PAck64#	Bidi PCI
AE5	PPerr#	Bidi PCI
AF5	PSerr#	Output PCI
T3	PIntA#	Bidi PCI
AE13	PCBE[7]	Bidi PCI
AF15	PCBE[6]	Bidi PCI
AD15	PCBE[5]	Bidi PCI
AF16	PCBE[4]	Bidi PCI
Y2	PCBE[3]	Bidi PCI
AC1	PCBE[2]	Bidi PCI
AE6	PCBE[1]	Bidi PCI
AE8	PCBE[0]	Bidi PCI
AE16	PAD[63]	Bidi PCI
AD17	PAD[62]	Bidi PCI
AF17	PAD[61]	Bidi PCI
AC17	PAD[60]	Bidi PCI

Pin#	Pin Name	Type
AE17	PAD[59]	Bidi PCI
AE18	PAD[58]	Bidi PCI
AF18	PAD[57]	Bidi PCI
AD18	PAD[56]	Bidi PCI
AE19	PAD[55]	Bidi PCI
AC19	PAD[54]	Bidi PCI
AD19	PAD[53]	Bidi PCI
AD20	PAD[52]	Bidi PCI
AF20	PAD[51]	Bidi PCI
AC20	PAD[50]	Bidi PCI
AE20	PAD[49]	Bidi PCI
AD21	PAD[48]	Bidi PCI
AE21	PAD[47]	Bidi PCI
AE22	PAD[46]	Bidi PCI
AF22	PAD[45]	Bidi PCI
AD22	PAD[44]	Bidi PCI
AF23	PAD[43]	Bidi PCI
AE23	PAD[42]	Bidi PCI
AE24	PAD[41]	Bidi PCI
AD26	PAD[40]	Bidi PCI
AC26	PAD[39]	Bidi PCI
AC24	PAD[38]	Bidi PCI
AC25	PAD[37]	Bidi PCI
AB24	PAD[36]	Bidi PCI
AB26	PAD[35]	Bidi PCI
AB25	PAD[34]	Bidi PCI
AA25	PAD[33]	Bidi PCI
AA24	PAD[32]	Bidi PCI
U2	PAD[31]	Bidi PCI
V2	PAD[30]	Bidi PCI

Pin#	Pin Name	Type
V1	PAD[29]	Bidi PCI
V3	PAD[28]	Bidi PCI
W1	PAD[27]	Bidi PCI
W3	PAD[26]	Bidi PCI
W2	PAD[25]	Bidi PCI
W4	PAD[24]	Bidi PCI
Y3	PAD[23]	Bidi PCI
AA2	PAD[22]	Bidi PCI
AA1	PAD[21]	Bidi PCI
AB3	PAD[20]	Bidi PCI
AB1	PAD[19]	Bidi PCI
AC3	PAD[18]	Bidi PCI
AB2	PAD[17]	Bidi PCI
AC2	PAD[16]	Bidi PCI
AD6	PAD[15]	Bidi PCI
AE7	PAD[14]	Bidi PCI
AC7	PAD[13]	Bidi PCI
AF7	PAD[12]	Bidi PCI
AD7	PAD[11]	Bidi PCI
AF8	PAD[10]	Bidi PCI
AD8	PAD[09]	Bidi PCI
AE9	PAD[08]	Bidi PCI
AE10	PAD[07]	Bidi PCI
AC9	PAD[06]	Bidi PCI
AF10	PAD[05]	Bidi PCI
AC10	PAD[04]	Bidi PCI
AE11	PAD[03]	Bidi PCI
AD10	PAD[02]	Bidi PCI
AF11	PAD[01]	Bidi PCI
AD11	PAD[00]	Bidi PCI
B15	LClk	Input TTL
D26	LClk_1X	Input TTL PD*
A15	PLLVSS	Input Ground for PLL
C16	PLLVDD	Input Power for PLL
E23	PLLen	Input TTL PU†
C21	Loe#[1] / Lsoe#	Output 12mA
B21	Loe#[0] / Lsce#	Output 12mA
B20	Lwe#[1]	Output 12mA
A20	Lwe#[0] / LsADSC#	Output 12mA
D18	Lce#[8]	Output 4mA

Pin#	Pin Name	Type
C18	Lce#[7] / Lsbw#[7]	Output 4mA
B18	Lce#[6] / Lsbw#[6]	Output 4mA
A18	Lce#[5] / Lsbw#[5]	Output 4mA
C19	Lce#[4] / Lsbw#[4]	Output 4mA
B19	Lce#[3] / Lsbw#[3]	Output 4mA
A19	Lce#[2] / Lsbw#[2]	Output 4mA
D20	Lce#[1] / Lsbw#[1]	Output 4mA
C20	Lce#[0] / Lsbw#[0]	Output 4mA
C11	LAddr[22]	Output 12mA
B11	LAddr[21]	Output 12mA
A11	LAddr[20]	Output 12mA
D12	LAddr[19]	Output 12mA
C12	LAddr[18]	Output 12mA
B12	LAddr[17]	Output 12mA
A12	LAddr[16]	Output 12mA
D13	LAddr[15]	Output 12mA
C13	LAddr[14]	Output 12mA
B13	LAddr[13]	Output 12mA
A13	LAddr[12]	Output 12mA
C14	LAddr[11]	Output 12mA
A14	LAddr[10]	Output 12mA
D15	LAddr[09]	Output 12mA
C15	LAddr[08]	Output 12mA
B16	LAddr[07]	Output 12mA
D17	LAddr[06]	Output 12mA
C17	LAddr[05]	Output 12mA
B17	LAddr[04]	Output 12mA
A17	LAddr[03]	Output 12mA
A21	LDat[63]	Bidi 4mA PD
D22	LDat[62]	Bidi 4mA PD
C22	LDat[61]	Bidi 4mA PD
B22	LDat[60]	Bidi 4mA PD
A22	LDat[59]	Bidi 4mA PD
C23	LDat[58]	Bidi 4mA PD
A23	LDat[57]	Bidi 4mA PD
D25	LDat[56]	Bidi 4mA PD
E25	LDat[55]	Bidi 4mA PD
E26	LDat[54]	Bidi 4mA PD
F24	LDat[53]	Bidi 4mA PD
F25	LDat[52]	Bidi 4mA PD

Pin#	Pin Name	Type
F26	LDat[51]	Bidi 4mA PD
G23	LDat[50]	Bidi 4mA PD
G24	LDat[49]	Bidi 4mA PD
G25	LDat[48]	Bidi 4mA PD
G26	LDat[47]	Bidi 4mA PD
H23	LDat[46]	Bidi 4mA PD
H24	LDat[45]	Bidi 4mA PD
H25	LDat[44]	Bidi 4mA PD
H26	LDat[43]	Bidi 4mA PD
J24	LDat[42]	Bidi 4mA PD
J25	LDat[41]	Bidi 4mA PD
J26	LDat[40]	Bidi 4mA PD
K23	LDat[39]	Bidi 4mA PD
K24	LDat[38]	Bidi 4mA PD
K25	LDat[37]	Bidi 4mA PD
K26	LDat[36]	Bidi 4mA PD
L24	LDat[35]	Bidi 4mA PD
L25	LDat[34]	Bidi 4mA PD
L26	LDat[33]	Bidi 4mA PD
M23	LDat[32]	Bidi 4mA PD
M24	LDat[31]	Bidi 4mA PD
M25	LDat[30]	Bidi 4mA PD
M26	LDat[29]	Bidi 4mA PD
N23	LDat[28]	Bidi 4mA PD
M24	LDat[27]	Bidi 4mA PD
N25	LDat[26]	Bidi 4mA PD
N26	LDat[25]	Bidi 4mA PD
P24	LDat[24]	Bidi 4mA PD
P25	LDat[23]	Bidi 4mA PD
P26	LDat[22]	Bidi 4mA PD
R23	LDat[21]	Bidi 4mA PD
R24	LDat[20]	Bidi 4mA PD
R25	LDat[19]	Bidi 4mA PD
R26	LDat[18]	Bidi 4mA PD
T24	LDat[17]	Bidi 4mA PD
T25	LDat[16]	Bidi 4mA PD
T26	LDat[15]	Bidi 4mA PD
U23	LDat[14]	Bidi 4mA PD
U24	LDat[13]	Bidi 4mA PD
U25	LDat[12]	Bidi 4mA PD

Pin#	Pin Name	Type
U26	LDat[11]	Bidi 4mA PD
V23	LDat[10]	Bidi 4mA PD
V24	LDat[09]	Bidi 4mA PD
V25	LDat[08]	Bidi 4mA PD
V26	LDat[07]	Bidi 4mA PD
W24	LDat[06]	Bidi 4mA PD
W25	LDat[05]	Bidi 4mA PD
W26	LDat[04]	Bidi 4mA PD
Y24	LDat[03]	Bidi 4mA PD
Y25	LDat[02]	Bidi 4mA PD
Y26	LDat[01]	Bidi 4mA PD
AA26	LDat[00]	Bidi 4mA PD
D8	LED_Traffic	Output 12MA slew
B7	LED_Link	Output 12MA slew
A10	UART_in	Input TTL PD
B10	UART_out	Output 4mA slew
C10	EE_Clk	Output 4mA slew
D10	EE_Dat	Bidi 4mA PD slew
B8	Misc[2]	Output 12mA
A8	Misc[1]	Bidi 4mA PD
C9	Misc[0]	Bidi 4mA PD
C5	JTCK	Input TTL
A5	JTRST	Input TTL PD
C7	JTMS	Input TTL PU
B5	JTDI	Input TTL PU
C6	JTDO	Output 4mA
B4	PROCMON	Output 4mA
A4	IDDTN	Input TTL PD
D5	TN	Input TTL PU
A7	TClk_in	Input TTL
M1	TClk_out	Output
B6	RClk0 / MII_TClk	Input TTL
A6	RClk1	Input TTL
A9	MII_Clk	Output 4mA slew
B9	MII_Dat	Bidi 4mA PD slew
M2	LnkRdy	Input TTL PD
M4	COL	Input TTL PD
M3	CRS	Input TTL PD
R3	TDat[9] / Tx_Error	Output 4mA
R2	TDat[8] / Tx_Enable	Output 4mA

Pin#	Pin Name	Type
R1	TDat[7]	Output 4mA
P4	TDat[6]	Output 4mA
P3	TDat[5]	Output 4mA
P2	TDat[4]	Output 4mA
P1	TDat[3]	Output 4mA
N3	TDat[2]	Output 4mA
N2	TDat[1]	Output 4mA
N1	TDat[0]	Output 4mA
J3	RDat[9] / Rx_Error	Input TTL PD
J2	RDat[8] / Rx_DatVal	Input TTL PD
J1	RDat[7]	Input TTL PD
K4	RDat[6]	Input TTL PD
K3	RDat[5]	Input TTL PD
K2	RDat[4] / MII_Link	Input TTL PD
K1	RDat[3]	Input TTL PD
L3	RDat[2]	Input TTL PD
L2	RDat[1]	Input TTL PD
L1	RDat[0]	Input TTL PD

Pin#	Pin Name	Type
F1	txp	Output differential
G2	txn	Output differential
F2	rxp	Input differential
E1	rxn	Input differential
C2	bgvss	Input ground
J4	gdvss	Input ground
E3	rxbvss	Input ground
E4	rxvss	Input ground
G1	txbvss	Input ground
H2	txvss	Input ground
D3	bgvdd	Input power
G4	rxbvdd	Input power
E2	rxvdd	Input power
F3	txbvdd	Input power
G3	txvdd	Input power
C1	iref	Input analog
D2	termres	Input analog
D1	vtxlo	Input analog

* PD = Light Pull Down

† PU = Light Pull Up

NOTE: *The bit-numbering convention is different for SERDES components. The transmit and receive data paths may need to be bit-reversed when connected to these components. The Tigon assumes bit-9 is the first bit transmitted and received when in external SERDES mode.*

Appendix A Tigon II (Revision 6) Errata

DMA 4

Summary:

When using a local synchronous SRAM memory system and either DMA channel is saving a checksum result at the end of a DMA, the checksum is written to the correct memory location in addition to being written to the local address field of the current Read channel DMA descriptor. This can also cause read DMAs to complete without error having transferred the data to the corrupted local address location.

Details:

The DMA assist logic manages the process of starting the DMA channels based on DMA descriptors located in local memory. The assist logic may optionally also save the final checksum of a DMA to a local memory location. When using synchronous SRAM, the SRAM arbitration logic interprets this checksum write as a request for two writes. It just so happens that the address used during the second “bogus” write is the local address field (word 2) of the current read channel DMA descriptor.

It is possible for the second write to not occur if after the first write a write request is processed the SRAM arbitration logic receives a write request from either of the read DMA channel or the MAC receive interface. Since this can never be guaranteed, it is safe to say that most of the time the second write will occur.

If only the read DMA channel saved checksums, then the local address field in the descriptor would get corrupted only after the DMA had completed. This has no adverse effect on the hardware, but may affect the firmware if it needs to rely on the local address field remaining unchanged,

Another problem is if the write DMA channel saves checksums, since it can corrupt the local address field of the read DMA channel. If the read DMA had already started then the corruption would not be a problem for the hardware. If the read DMA had not started, then the read DMA would use the corrupted local address field when the DMA starts.

The most serious scenario is that it is possible for the “bogus” checksum write from one DMA channel to screw up the host address read for the other DMA channel if the read immediately follows the write. This results in a 64-bit host address which has the 16-bit checksum result repeated every 16-bits. The host address field in the descriptor will not be affected, but the local address field will receive the “bogus” checksum write. It is not possible for a checksum write to corrupt the host address of the next DMA for the same DMA channel.

The following conditions must all be met in order to provide the environment which causes the error.

1) Using synchronous SRAM for local memory.
2) The Save Checksum bit is being set in either Read or Write DMA channels.

Work-around Options:

1) Use asynchronous local memory.

2) Do not set the Save Checksum bit in both DMA channels.

3) Use the DMA assist logic to manage only the read channel with checksum saving enabled on selected or all read DMAs. Use firmware to manage the write channel and perform the tasks the assist logic would normally do. A shadow write channel descriptor producer will be needed since the assist logic must always see an empty write ring. After firmware starts a write DMA, it must advance the assist producer by one entry so that when the DMA finishes and the assist increments the consumer, the ring will look empty to the assist logic. As part of initialization, the assist logic should be allowed to start one write DMA (can be a “dummy” DMA).

4) Same as work-around 3 except swap the read and write DMA channels. Keep in mind the write channel can alter the local address of the read descriptor. Either move the firmware read descriptor region or ensure the read consumer points to an unused descriptor.

5) Completely disable the assist logic and use firmware to emulate the proper behavior of the assist logic. The assist producer, consumer, and reference can still be used to generate events. Keep in mind that the assist logic is also responsible for updating the transmit buffer producer at the end of a read DMA if instructed to do so in the read DMA state register.

MAC 6

Summary:

This error affects the MAC running at any speed and can cause receive frames which have been truncated to be reported as good. The receive descriptor's start address will be correct, but the length and status word will be that from the next packet which gets received into memory.

Details:

There is a state machine which places the packet data followed by the descriptor into the FIFO which goes between the MAC and the local memory. This state machine will abort any frame if the receive buffer completely fills so as not to overrun the local memory buffer.

The MAC provides all of the packet data and then waits one clock cycle to signal if the CRC was correct and the packet is good. If the buffer fills such that during the one cycle after the MAC has provided the last bytes of a packet but before the CRC is checked the buffer full indication arrives, the packet's ending status word never gets written into the FIFO. The local memory interface will wind up concatenating this packet with the next packet received after buffer space becomes available.

If the MAC is in half-duplex gigabit mode and receives a packet which is being carrier extended, then the one clock cycle mentioned above will be expanded until after carrier extension has ended. However during carrier extension, no additional data bytes are being received so the probability of the receive buffer going full at that time is not significantly increased.

In the receive buffer, the packet will look complete with the possibility that between 0 and 15 bytes (on double-word boundaries) can be truncated.

If the keep length from the receive state register has not been exceeded, then no error will be generated. Packets with lengths up to 16 byte longer than the keep length (i.e. up to next modulo 16 length) can arrive without error. For example if the keep length is set to 64 bytes, then packets up to 79 bytes will not have this error.

The hardware updated MAC statistics are updated according to the actual number of received packets and are not affected by this error condition.

Work-around Options:

1) Use the 802.3x flow-control to avoid completely filling the receive buffer.
2) Set the keep length to 64 bytes to avoid the error for packets up to 79 bytes in length.
3) With synchronous local memory, it is highly unlikely to get this error with packets whose length can be contained in an even number of double words in local memory.
4) Prior to processing a receive descriptor, if the number of bytes between the buffer starting address of this descriptor and the buffer starting address of the next descriptor (if available), is greater than or equal to the first descriptor's length + 80 bytes, then there is very high probability that the first descriptor should be thrown away.
5) Similar to work-around 4, except that there is no need to do such checking until the buffer has the possibility of becoming full. If the receive buffer low attention is never received, then no descriptor checking need be done. This may provide a possible performance optimization.

CPU 3

Summary:

When using synchronous SRAM, the result of a load instruction from external SRAM which was not in the data cache will return an incorrect value which happens to equal to the value from a recent external SRAM write.

Impact:

This error requires a particular deterministic sequence of four instructions. Since the data cache is of marginal value it will be easier to merely disable the data cache in most applications. If it is to be enabled, then the code should be checked for the offending sequence.

Details:

The data cache has snooping logic to keep it consistent which external SRAM. The snooping logic has an error which can cause the second of two back-to-back snoop hits to update the cache just after the cache had been cleared due to a new read from the CPU to a non-cached location. This may results in an incorrect value being given to the CPU in response to a external SRAM read.

The following instruction sequence must be met in order to cause the problem:

1) A store to external SRAM to an address which is valid in the data cache.
2) A second store to external SRAM to an address which is valid in the data cache.
3) A store to any address external to the CPU except that which is in the data cache or scratch pad.
4) A load from an external SRAM address not in the data cache.

This code sequence is not sufficient to guarantee the error since it is vital that the SRAM arbiter group the first two writes together in back-to-back operations. This can only happen with synchronous SRAM. If any of the four instructions use the scratch pad, then the error will not occur.

This error was observed at the start of a routine which was pushing a bunch of entries on the stack prior to doing the first load operation. The data cache happened to contain a portion of the stack since the end of the prior routine popped some entries off of the stack.

The following conditions must all be met in order to provide the environment which causes the error:

1) Using synchronous SRAM for local memory.
2) Data cache is enabled.
3) The above sequence of four instructions.

Work-around Options:

Any work-around would focus on avoiding any one of the conditions listed above, such as

1) Use asynchronous SRAM for local memory.

2) Disable data cache.

3) Only enable data cache when needed. Verify portion of code with data cache enabled does not meet criteria. If it does, then either swap the last 2 stores, or re-write source code to generate different instruction sequence.

4) Since this code is most likely to only get generated at the start of routines which push entries onto the stack, consider placing the stack in the internal scratch pad.