

Z X - 7 P i r a n h a M a i n B o a r d
H a r d w a r e S p e c i f i c a t i o n

Revi si on A
10/03/89
Mark D. Ni col

***** CONFIDENTIAL, FOR ZDS INTERNAL USE ONLY *****

THIS DOCUMENT CONTAINS PROPRIETARY AND CONFIDENTIAL INFORMATION. IT IS INTENDED ONLY FOR LIMITED DISTRIBUTION WITHIN ZENITH DATA SYSTEMS AND ZENITH ELECTRONICS CORPORATION.

1. Introduction

1.1 Scope

This document is intended to present the hardware specification for the ZX-7 series *Piranha* main board. The main board circuitry consists of the CPU, numeric coprocessors, memory, EISA bus controller, EISA bus interface, cache controller, cache interface, and EISA expansion slots.

1.2 System Overview

The Piranha main board is the primary component in a high performance, 32-bit, multi-tasking, multi-user EISA compatible desktop personal computer. It's design employs state of the art technology, including high speed microprocessors, custom gate arrays, and surface mount components. Key design features include an operating speed of 33 megahertz, with a high speed cache memory and a 16 word deep write queue, which together combine to provide near zero wait state performance.

System expansion is through an industry standard EISA compatible bus which provides compatibility with existing 16-bit and 8-bit hardware and software, and provides extensions to enable 32 devices to be utilized.

1.3 Specifications

CPU:	Intel 80386
Clock Speed:	33 MHz
Coprocessor Option:	Intel 80387 Weitek WTL 3167
Memory, Standard:	4 Megabytes
Max. (1M SIMM):	8 Megabytes
Max. (4M SIMM):	32 Megabytes
Max Addressable:	4 Gigabytes
Cache:	16K x 32 direct mapped (optional) 4K x 32 direct mapped (standard) 2K x 32 direct mapped (optional)
Cache hit rate:	95 per cent or greater
Write Queue:	16 locations deep x 66 bits wide
System Expansion:	8 EISA compatible slots
Circuit Board:	8 layers, SMT and thru-hole

2. Design Constraints

Because of a limitation of the system bus controller the data is not always made available to the cache subsystem when a bus snoop occurs. This limits the cacheability range of the memory to local system memory only, bus memory cannot be cached. This is necessary to maintain cache coherency.

To reach market quickly, an existing memory controller design was used, which does not exhibit the speed necessary to allow EISA *burst* mode memory cycles.

Another limitation of the existing memory controller is that only 256K of hardware EMS can be directly supported.

One design objective of the Piranha was to limit or eliminate the use of hardware jumpers or dip switches. This was accomplished through software programmability of the custom gate arrays.

The system chassis is identical with the existing Mongoose box, which somewhat restricts the amount of on-board memory and expansion capability.

3. Hardware Functionality

The backplane/main board contains all the circuitry for the Piranha system with the following exceptions:

- * I/O board
 - Real Time Clock w/battery backup
 - System ROM
 - System Control Processor
 - Serial & parallel I/O
- * Disk controller board
- * Video system board
- * Cache sub-system (does not require a bus slot)

Additional components on the main board include a standard DIN connector for the detachable keyboard, an input power connector, an auxiliary power connector for a second fan, and six power supply status light emitting diodes for ACOK, DCOK, +12V, -12V, +5V, and -5V.

The 644-58 system/cache controller, which interfaces directly to the 644-57 memory controller and the 644-61 posted write queues, with an 80386 and a numeric coprocessor form the heart of this state of the art computing engine.

The following sections will describe in limited detail the individual subsections of the Piranha main board. Refer to figure 1.0 for a block diagram, and figure 2.0 for a component layout of the main board.

The BLOCK Diagram of the system goes here.

The Component Layout Diagram goes here.

3.1 CPU

The Central Processing Unit (CPU) of the Piranha is the Intel 80386. This is the third generation of the 80x86 family and provides a true 32-bit internal and external architecture. The 80386 instruction set is a superset of the earlier 8086 and 80286 CPU's, thereby making it compatible with existing software, yet offering a significant performance increase. Additional advantages of the 80386 are dual addressing modes (real and protected), very large addressing space (4G physical, 64T virtual), on-chip memory management unit featuring virtual memory support, paging, and four levels of protection.

The CPU is capable of running in a *pipelined* mode to remove one wait-state in slower memory designs. This mode is not used in the Piranha since it complicates system timing, and the high speed cache memory design typically runs with zero wait states already.

A software programmable *slow mode* is provided in the hardware to allow the use of certain speed critical or copy protected software written for slower computers. This mode simply slows the CPU to approximately the speed of an 8MHz 1 wait-state AT class computer.

For information on reference material for the 80386 CPU, see appendix C.

3.2 CoProcessors

To handle high speed numeric processing, the Piranha provides two co-processor options, the Intel 80387, and the WEITEK WTL3167. Accommodation for both chips is provided for simultaneously, using the two sockets provided on the board. The 80387 provides for industry standard high speed numeric processing, while the WEITEK provides for a higher performance system.

3.2.1 80387

The Intel 80387 is the latest generation of the 80x87 family and provides a true 32-bit internal and external architecture. The 80387 instruction set is a superset of the earlier 8087 and 80287 NPU's, thereby making it compatible with existing software, yet offering a significant performance increase.

For information on reference material for the 80387, see appendix C.

3.2.2 WEI TEK

The WTL 3167 is a single chip CMDS implementation of the board level WTL 1167 numeric co-processor. It is not software compatible with the Intel 80x87 family, but does provide 3 to 4 times the performance of the 80387 for speed critical tasks such as CAE/CAD. Many compute intensive software packages already provide support for the chip.

The WTL 3167 coprocessor is a memory-mapped peripheral, meaning that to the 80386 and it's application software, the WTL 3167 appears to be a segment of memory at an upper address. Instructions are executed by performing memory moves to and from the coprocessor.

For information on reference material for the WEITEK 3167, see appendix C.

3.3 Cache

To provide the highest level of performance available, the system CPU must run with as close to zero-wait states as possible. However, the Piranha's fast clock speed presents severe design problems for a conventional memory design. For this reason a high speed cache memory has been implemented.

The function of a cache memory system is to provide fast local storage of frequently accessed code and data. The cache system intercepts 80386 memory references to see if the required data resides in the cache. If the data resides in the cache (a hit), it is returned to the CPU without incurring additional wait-states. If the data is not cached (a miss), the reference is forwarded to the system and the data is retrieved from main memory.

An efficient cache will yield a high "hit rate" (the ratio of cache hits to total 80386 accesses), such that most accesses are serviced with zero wait states. The net result is that the wait states incurred by the relatively infrequent miss are averaged over a large number of accesses, resulting in an average of nearly zero wait states per access.

A good hit rate is an essential ingredient of a successful cache implementation, but it is not the only factor for performance consideration. Just as essential are sound cache management policies, including the handling of CPU writes and the preservation of cache coherency.

Another true measure of the success of a cache design is it's ability to meet an individual system cost/performance requirement. Some systems require only a moderate jump in performance while affecting only a minimal cost impact on the overall system. Other systems will want to squeeze every last percent of performance out of an architecture despite the cost. The Piranha, with it's *adaptable architecture*, can easily be tailored to nearly any cost/performance specification.

The Piranha cache subsystem is controlled by a custom gate array that resides between the CPU and the rest of the system, and appears to be the CPU to the rest of the system.

The cache memory itself resides on a plug-in card, which not only reduces the required main board area, but also accommodates the *adaptable architecture* philosophy. For example, the cache can be organized as a 16K direct mapped cache, which is the standard configuration, or by simply changing the plug-in card, a 64K two way set associative cache can be implemented.

3.3.1 Cache Controller

The Piranha system/cache controller is incorporated in a custom BiCMOS gate array (644-58). This was required due to the complexity of the design and the limited main board space available.

The main function of the system/cache controller is to interface the CPU with the memory subsystems, using either the high speed cache, or the lower speed system memory. However, as it's name implies, the controller contains more than just the cache control logic. It also contains bus arbitration logic, write queue interface control logic, bus interface control logic, bus snoop control logic and processor and coprocessor interface and reset logic.

The system/cache controller is designed to interface directly to the Intel 80386 CPU forming a *local* processor bus. The local bus interface includes the numeric coprocessor and the cache memory subsystem as well as the CPU itself.

The cache memory subsystem interfaces directly to the cache controller. This design makes extensive use of pipelining within each cache cycle to allow for speedups that would otherwise not be possible. The data buffers/latches shown between the processor and the cache memory subsystem are directly controlled by the cache controller. These 74F543 type devices with their latching as well as buffering capability allow the cycle pipelining to take place.

The system/cache controller interfaces to the remaining system hardware through its *system* bus. The system bus interface appears essentially as the 80386 to the system bus. The system bus interface on the system/cache controller includes direct interface to the memory/bus controller. The system bus also interfaces directly to the high performance posted write queue.

When the CPU performs a memory write cycle, a system memory access must occur, since a cache write alone will most likely cause the data to be lost if that location of the cache is subsequently overwritten. This system memory access would normally cause the CPU to slow down substantially since system memory cannot respond without adding additional wait- states.

The Piranha potentially eliminates this performance robbing situation by latching all CPU address, data, and control lines into the write queues, and allowing the CPU to continue with the next instruction. This allows the bus/memory controller to complete the write to system memory concurrently and asynchronously. This *write posting* allows up to sixteen write cycles to be queued to the bus controller before any delay is passed on to the CPU. The posting of writes to the system bus allows them to occur in zero wait states on the local bus, though they may take much longer to complete on the system bus.

When the CPU requests data, the cache controller first checks to see if the requested data is cached. If so, the data is immediately made available and the cycle terminated. If the data is not cached the controller must pass the request on to the memory controller, or the bus controller.

The cache controller will latch all the address, data, and control lines from the CPU into the write queue and assert `DOCYCLE`. Further processing by the CPU is suspended until the write queue is flushed of any previous write cycles that were still pending and the read cycle can be completed.

If the requested data resides in the local system memory, the memory controller will initiate a local memory cycle, generating all the necessary commands to access the data. If the data is not located in the local memory, then the request is passed on to the system bus controller.

The bus controller determines if the requested data resides on an 8-bit, 16-bit, or 32-bit device and issues the appropriate cycles to read a complete double word, since the cache's minimum transfer (line) size is 32-bits. When the 32-bit data is made available the cache controller latches it back into the write queues, delivers it to the CPU, updates the cache, and terminates the cycle.

The system bus arbitration is handled directly by the cache controller freeing the CPU from this task. Since the CPU does not control the system bus arbitration it never needs to enter the hold state. The performance benefit of this is that it allows parallel hardware operation between the local bus and the system bus.

This hardware parallelism is used several different ways. When the system bus has been given away for refresh, DMA or temporary MASTER accesses the CPU may continue running on the local bus, through read or write accesses to the cache, the cache controller registers, or the numeric coprocessor. The pipelining of queued writes to the queue is also one of the factors allowing the parallel bus operation. This parallel operation can continue on both busses up to the point when the CPU must make a system bus read access, or a posted write operation occurs when the write queue is already full. Then the parallel operation stops and the CPU waits, idling on the local bus until the system bus is returned to it. This parallelism allows *pseudo hidden* refresh, DMA and temporary master accesses.

The parallel operation of the system bus and the local bus means that CPU accesses on the local bus can occur simultaneously as temporary master accesses on the system bus. This parallel operation greatly enhances performance since the two separate busses seldom have to wait for each other to complete. There is a problem with the parallel operating busses however. The problem is *cache coherency*.

When both busses are operating simultaneously and the system bus is performing a write to a main memory location that is currently contained in the cache, the cache must be updated to make it's data match with main memory. The cache controller directly controls this function through an operation called *bus snooping*.

Bus snooping is the means by which the system/cache controller watches the bus activity at all times, regardless of what other tasks that it or the CPU are involved in. Bus snooping functions by watching the system bus write lines. Whenever a system bus write occurs the cache controller controls the latching of the current bus address and data values into external 74F573 type devices. CPU accesses to the cache are then interrupted, if they are occurring, and the address of the system bus write is checked in the cache to see if that entry is in the cache. If the snoop check finds that the entry was not in the cache than normal operation resumes. If the snoop check finds that the entry was in the cache, then that cache data entry is updated and normal operation can resume. The updating of the cache data entries through bus snooping maintains cache coherency and boosts system performance by allowing the parallel operation of the system and local bus to continue.

The cache controller also controls the reset to both the CPU and the numeric coprocessor. The numeric coprocessor is only reset at power up time. The CPU is reset at power up time and also can be reset individually at any time through the use of the RC input to the cache controller. The reset pulse to the CPU from the cache controller will always be a minimum of 16 cycles of CLK2. The length of the numeric coprocessor reset will be as long as the power up reset pulse.

The local bus, or processor ready (-pRDY) signal is controlled solely by the system/cache controller. It integrates the ready signals from all possible cycles. These include accesses to the coprocessor, to the internal cache controller registers, to the cache data RAM or accesses to the system bus.

3.3.2 Cache Controller Programmability

The system/cache controller is software programmable to configure or report on various aspects of the cache system. The I/O ports are defined as follows:

CMCR - *Cacheable Memory Control Register*

Used to program the areas of cacheable system memory.

CSSR - *Cacheable System Status Register*

Used to program miscellaneous status inputs.

CTTR - *Cache TAG Test Register*

Used to read the status of the cache tag match bits.

CIDR - *Cache Identification Register*

Used to read a four bit cache ID code containing the cache size and organization.

These ports are decoded externally of the controller itself to allow flexibility when used in other systems. The actual addresses are detailed in section 7.

Accommodations are provided to extend the address range of the cache controller or achieve finer granularity than that provided by the programmable registers. This is done by populating an address decoder PAL and driving the -NCC0 and/or the -CC0 lines.

See appendix C for additional information on the system/cache controller.

3.3.3 Cache Timing

The advantage of a cache memory system would be severely compromised if the cache system was unable to respond without incurring additional memory wait-states. The 80386 is capable of executing a memory access in as little as two processor clock cycles, which at 33MHz clock rates translates to a window of only 60 nanoseconds. Clearly, the timing of the cache subsystem is extremely critical to the overall performance of the entire system.

3.3.4 Cache Interface

The Piranha cache interface consists of latches and buffers for holding and multiplexing address and data, controlled by the system/cache controller. The cache card is mounted on a dedicated 100 pin connector allowing for different sizes and configurations of cache to be easily connected. The cache connector contains a 4-bit cache ID which, when driven by the different versions of cache cards, signal to the cache controller which version of cache memory is installed.

Section 3.3.5 defines the 4-bit ID codes and figure 4.0 specifies the cache connector pinout.

3.3.5 Cache Options

The plug-in configuration of the cache memory system and the *adaptable architecture* of the system/cache controller allow extreme flexibility in the structure of the cache subsystem. Different memory sizes and organizations allow the cache subsystem to be tailored to specific performance needs or applications. With these choices almost every cost/performance goal will be attainable.

The specific cache sizes and organizations specified to date are:

ID#	Size x 32	Organization	Total memory
F	2K	<i>direct mapped</i>	8K
D	4K	<i>direct mapped</i>	16K
9	16K	<i>direct mapped</i>	64K
0		<i>no cache installed</i>	

In addition to the direct mapped organization currently implemented, the cache controller also can support two way set associative caches. The two way set associative cache can be thought of as two direct mapped caches in parallel. The performance advantage over a direct mapped cache is that all identical page offsets map to two cache locations instead of one, reducing the potential for "*thrashing*".

Thrashing occurs when two or more data items want to occupy the same location in cache, and they repeatedly overwrite each other causing a hit rate of zero.

The two way set associative cache can be more efficient and result in a slightly higher hit rate in some situations. For instance, this option may become more attractive in a *UNIX* type environment.

The cache connector pinout goes here.

3.4 Write Queue

The write queue is implemented as a matched trio of custom CMOS gate arrays (644-61). It is fundamentally a 66 bit wide, 16 location deep, transparent latch/buffer between the cached processor and the system/bus controller. It will bi-directionally buffer address, control signals, and data between these devices, and provide externally controlled cached processor write command posting. Control lines provide the ability to latch and enable data, addresses, and commands separately in either direction, and to force all byte enables active.

The write queue is a logical extension of the single *posted write buffer* scheme used by competitive cache controllers. With any *write through* cache architecture, such as the Piranha, all processor writes (cache hit or miss) must be broadcast to the system. Not only is the CPU slowed to the system bus speed during each write, but the system bus is also engaged by the CPU during each write.

An empty write queue will store up to 16 write commands (address, control, and data) issued by the cached processor without adding wait states, in a first-in, first-out (FIFO) format. When the queue is full, the processor has to wait until the first command that was posted has finished.

Studies have shown that software typically exhibits roughly 15 to 20 percent total write cycles, and that many writes occur in bursts of more than one. In a typical system, most of the consecutive writes will occur during block moves and stack manipulations. Block instructions, such as programmed disk IO, are usually infrequent but rather lengthy. These operations will quickly fill the queue, causing the CPU to wait. Stack manipulations occur during hardware and software initiated interrupts, and during high level language procedural calls. These operations will generally cause at least two to three words to be pushed to the program stack. If these locations have already been cached, the queue should never fill, allowing the CPU to continue processing while the memory writes are handled concurrently by the bus/memory controller. As long as all the queued writes can be processed by the bus/memory controller before any cache misses occur, the CPU will incur no additional wait-states.

If a cache miss should occur immediately following a queued write, the benefit of having posted the write is voided, since the queue must be flushed and the CPU halted until the cache miss can be serviced. This is necessary since the queue contents may affect the data to be retrieved. In addition, the bus cannot be granted to a temporary MASTER if the queue is currently in the middle of a locked cycle. This will usually occur during a semaphore Read-Modify-Write (RMW) sequence when the location is a cache hit.

3.5 Memory

The system memory of the Piranha is organized as 8 banks of either 1 or 4 megabyte *Single In-line Memory Modules* (SIMM s). These modules include 32 bits of data and 4 parity bits, one for each of the 4 bytes. The banks are all socketed and are easily field upgradeable.

The memory controller gate array provides all the necessary control signals to read, write, refresh, and error check the system memory. The design of the controller allows the memory to be intermixed with 1 megabyte and 4 megabyte SIMM s in any combination. Any additional memory, up to the maximum of 4 gigabytes, must reside in the memory expansion bus slots.

Programmable registers within the bus/memory controller allow each bank of memory to be dynamically sized and mapped. The first bank of memory on the main board also can be programmed to respond to *extended* or *EMS* accesses.

3.5.1 Memory Controller

All of the logic and control for the Piranha local memory subsystem is contained in one custom BiCMOS gate array. This array, which is ZDS part number 644-57, is packaged in a 136 pin quad flat pack.

The memory controller contains programmable registers to configure the main board memory. Each bank has a register that allows it's starting address and depth to be programmed. These registers allow 1 megabyte banks to be placed on any 1M boundary, and 4 megabyte banks to be placed on a 4M boundary.

On power-up, the first bank will be hardware programmed with a depth of 1 megabyte starting at 000000H, and all other banks will be disabled. The ROM will then program the next bank's starting address and attempt to size any existing memory. If memory is found in the second bank it's depth is programmed and the process is repeated on the next bank, until all contiguous memory has been found.

The configuration of the first 1 megabyte is programmable as to how *base*, *EMS*, and *extended* memory is allocated. *Base* memory can be isolated as the first 256K, 512K, 640K or disabled completely. The reserved memory space from 0A0000H to 0DFFFFH, the *slushware* space from 0E0000 to 0FFFFFFF, and any base memory not used, can be mapped to *EMS* board 0, mapped to *extended*, or disabled. See section 6 for details of the system memory map, and appendix C for more information on the bus/memory controller.

System memory cycles are started and terminated under the direct control of the system/cache controller. It issues a DOCYCLE signal to indicate the start of a cycle to the system bus.

When the memory controller determines that the addressed data resides in the local memory, it immediately starts a standard 32-bit memory cycle. The specific memory bank requested is encoded in three bank-enable lines and externally decoded to generate the RAS and CAS signals, which together with the multiplexed address lines determine which location in memory is accessed.

The termination of a local memory cycle occurs when the memory controller issues a RDYBUS signal. This signal must be issued in the proper phase of the cycle with adequate setup and hold time from the rising CLK2 edge that represents the end of the cycle.

3.5.2 Memory Controller Programmability

The bus/memory controller is software programmable to configure or report on various aspects of the system. Since the memory controller registers are mapped to I/O locations which conflict with the EISA bus slot I/O ranges, they are externally protected by the slushware write enable mechanism of the Piranha ROM. That is, if slushware protection is disabled the I/O cycle will go to the memory controller only, otherwise it will be sent to the bus only. The I/O ports are defined as follows:

- SPRE** - *ScratchPad Ram Enable (PRIVY)*
Enables access to protected I/O ports and memory.
- FMMR** - *First Megabyte Mapping Register*
Contains the starting address of any extra memory that is being mapped to extended.
- FMCR** - *First Megabyte Configuration Register*
Contains the amount of the first megabyte used as base, mapped to EMS, or mapped to extended.
- PEAR** - *Parity Error Address Register*
Contains information concerning the address of the last parity error experienced.
- 32SP** - *32-bit Speed Register*
- 16SP** - *16-bit Speed Register*
- 8SP** - *8-bit Speed Register*
Holds the number of wait-states (0-15) that will be added to the basic I/O and memory cycles.
- RISP** - *Refresh/INTA Speed Register*
Holds the number of wait-states (0-15) that will be added to refresh and interrupt acknowledge cycles.
- REVR** - *Revision Register*
Contains the revision number of the gate array.
- EMSO** - *EMS Page 0 Register*
Contains the EMS mapping register for EMS page 0.
- 0BCR** - *Bank 0 Configuration Register*
- 1BCR** - *Bank 1 Configuration Register*
- 2BCR** - *Bank 2 Configuration Register*
- 3BCR** - *Bank 3 Configuration Register*
- 4BCR** - *Bank 4 Configuration Register*
- 5BCR** - *Bank 5 Configuration Register*
- 6BCR** - *Bank 6 Configuration Register*
- 7BCR** - *Bank 7 Configuration Register*
Contains the starting address and memory depth for each of the 8 banks of main board memory.

3.5.3 Memory Cycle Timing

The local system memory located on the main board can be accessed with the addition of no 32-bit wait-states programmed into the bus/memory controller, for a total of four processor clock cycles from a valid DOCYCLE.

3.5.4 RAM modules

Single In-line Memory Modules (SIMM) are employed to house all 32-bit memory in the Piranha system since they are both convenient and economical of board space. The SIMM s are organized as 32-bit wide and either 256 kilobyte or 1 megabyte deep. This organization translates to either 1 megabyte or 4 megabytes of memory per bank. In addition each SIMM carries 4 bits of parity data, 1 bit for each of the 4 bytes. Use of the SIMM s allows for easy field upgrades or replacements.

The 1 megabyte SIMM s are packaged as multi-layer, surface mount PCB' s containing eight 256K x 4 DRAMs and four 256K x 1 DRAMs. The 4 megabyte SIMM s are similar, with the 256K parts being replaced by 1 megabyte parts. Interconnection to the main board is through a 72 pin card edge connector. The outline of the SIMM is shown in figure 4.0, however the component layout may differ.

3.5.5 Error Detection

All the local system memory on the Piranha main board is parity checked to detect single bit errors. For every byte written to memory, a bit is generated to produce odd parity, and is stored along with the data. When that byte is subsequently read from memory, it's parity bit is also retrieved. If upon examination, odd parity has not been maintained, a parity error is generated.

To test the parity circuitry, parity errors can be forced to occur by enabling the DISPG (disable parity) line. This causes even parity to be generated when writing to memory. Since odd parity is still used to check data read from memory, a parity error is generated, provided that the parity circuitry is performing correctly.

To report the error, the bus/memory controller contains an I/O port that will contain the six high order address lines, and the two low order address lines of the offending byte. The high order address lines decode which bank of memory contains the error, and the two low order address lines, decode which byte is at fault.

Since the system memory can be dynamically mapped and sized, the system can be made somewhat fault tolerant. If an error is detected during the power up sequence, that bank of memory can be disabled, and the next available bank mapped into it's place. This allows the user to continue on until that bank of memory can be repaired or replaced.

The SIMM module diagram goes here.

3.6 Expansion Bus

The expansion bus for the Piranha consists of eight industry standard EISA compatible slots. These can be used for the addition of extra memory beyond that provided for on the main board, and the interconnection of high speed 32-bit peripherals such as disk controllers and video boards.

The EISA 32-bit slots consist of the standard 62-pin PC connector outline, and the standard 36-pin AT connector outline, all in one special connector which doubles the amount of pins available. These extra pins are necessary for the upper 16 data bits, the additional address bits, four byte enable signals, and other signals required for system control.

One of the slots (P105) is physically reversed from the others and is reserved for use only by the system I/O board. This board is actually just an extension of the main board and is more fully described in another section. Each of the remaining expansion slots are identical, except for three signals. The SAENn signal is similar, but unique to each slot, to allow existing ISA boards and slot specific EISA boards to co-exist. The -sMREQn and -sMACKn signals are also unique to each slot except for slots P106 and P107 which share -sMREQ5 and -sMACK5. These signals are used by an EISA bus master device to gain bus ownership.

Refer to figure 2.0 for specific slot assignments and locations.

3.6.1 Bus Controller

Most of the logic and control for the Piranha bus is contained in the 82350 Intel chip set. This chip set comprises the 82358 *EISA Bus Controller* (EBC) and the 82357 *Integrated System Peripheral* (ISP). Together these two devices implement a functional interface to the EISA bus, and provide most of the standard peripheral functions necessary to implement an EISA solution.

The primary function performed by the EBC is the access and management of all data, either memory or I/O, which reside on the system expansion bus. This requires the generation of all command and control signals necessary to access data in any combination of 8, 16, 24, or 32 bits. The controller must generate the correct signals to build additional cycles and swap or reroute data paths when there is a data path width mismatch.

The monitoring of source and destination data path widths is accomplished using the AT signals M16, I16, HOLDA and MASTER16, and the EISA signals EX32, EX16, and EXMASTER. If, for example, the CPU performs a 32-bit read of an 8-bit PC style memory card, the bus controller will respond by building four consecutive PC type bus cycles, and routing the bytes from the lower 8-bit bus up to the appropriate 32-bit byte. This procedure does require substantially more time to perform than a 32-bit access to a 32-bit card, but is necessary to maintain compatibility with existing hardware.

3.6.2 Bus Timing

The timing for the expansion bus is designed to be fully compatible with the existing AT bus timing for all 8-bit and 16-bit cycles. The timing for the EISA extensions of the bus are available in the EISA bus specification available from BCPR Services Incorporated.

3.6.3 Signal Definitions

Signal definitions for the bus are identical to the EISA industry standard, which are defined in the BCPR specification. The pinout is shown in figure 5.0.

The Bus Pinout Diagram goes here

4. Mechanical

The Piranha main board is physically the same size as the existing MONGOOSE backplane, and resides in the same chassis.

5. Jumper Options

The Piranha was designed with the intention of reducing or eliminating the number of jumpers and switches needed to program the hardware. While this was considered successful, three unpopulated jumper locations still exist.

J100 and J101 are used when a system speed other than 33MHz is desired. In addition to the different speed oscillator needed, they also must be set to strap the EBC to generate the correct clock speed for the bus, as documented in the EBC data sheet.

J102 can be used to hide an 80387 coprocessor from the 80386 CPU. This jumper is primarily used for testing purposes only, to disable the 80387, without having to physically remove it.

6. Memory Map

The CPU can address 4 gigabytes of memory in its protected address mode. In real-address mode, it can address 1 megabyte of conventional (PC-compatible) memory. The first 640K of this range contains dynamic user memory. The remaining address range (up to 1 megabyte) is reserved for specific controllers (video, hard disk, etc.) and a window through which access to *expanded memory* (EMS) occurs. Access to memory beyond the CPU's 1 megabyte address range limit can only be accomplished through the 80386's protected address mode or the *expanded memory specification* (EMS) option.

The BIOS ROM on the I/O card is an evolution of the current Zenith AT ROM. The ROM in the Piranha copies itself into RAM at boot-up time, then disables writes to that area of RAM via the SCP. This is called *slushing* the BIOS or *slushware*. *Slushware* allows faster execution of BIOS calls, since the RAM can be accessed faster than the ROM. Also, because the 8-bit wide ROM is copied into the 32-bit wide cacheable system memory, only one ROM is required instead of four, resulting in space, power and cost savings. If there is an OEM ROM present, its code also will be *slushed*.

During coprocessor accesses, the system/cache controller will not pass the cycle out to the system, to avoid any unwanted response from the ROM or other device.

The following provides a complete map of the system's addressable memory. All addresses are in hexadecimal.

Address Range	Size	Description
C0000000-C1FFFFFF	32M	optional WEITEK coprocessor
*80000000-8003FFFF	256K	system ROM
:	:	:
<i>Top of addressable system memory</i>		
00100000-03FFFFFF	63M	<i>extended memory</i> (1M - 64M)
:	:	:
<i>Top of first megabyte of system memory</i>		
000F1000-000FFFFF	60K	System BIOS ROM <i>slushware</i>
000F0000-000F0FFF	4K	Scratchpad RAM
000E0000-000EFFFF	64K	Optional ROM <i>slushware</i>
000D0000-000DFFFF	64K	EMS window
000CA000-000CFFFF	24K	Open
000C8000-000C9FFF	8K	Hard disk BIOS ROM
000C0000-000C7FFF	32K	EGA video BIOS ROM
000B8000-000BFFFF	32K	CGA video RAM
000B0000-000B7FFF	32K	MDA video RAM
000A0000-000AFFFF	64K	EGA video RAM
00080000-0009FFFF	128K	System base memory (640K)
00040000-0007FFFF	256K	System base memory (512K)
00000000-0003FFFF	256K	System base memory (256K)

*duplicated every 256K to the 4 gigabyte top of memory except where the WEITEK coprocessor is decoded.

7. I/O Map

The following list defines the Piranha main board I/O port addresses, many of which are implemented in the memory controller. When an extended I/O address (beyond the standard 10 bits) is recognized by the memory controller or the system/cache controller, and it is decoded as an on-chip register, the cycle is not forwarded on to the rest of the system. This eliminates the possibility of a standard I/O device erroneously responding to an extended I/O request.

PORT	RW	DESCRIPTION
0000-000F	RW	DMA 1
0020	RW	INT 1 control register
0021	RW	mask register
0040	RW	Timer 1 Counter 0
0041	RW	Refresh Request
0042	RW	Speaker tone
0043	RW	Command mode register
0048	RW	Timer 2 Counter 0
0049	RW	reserved
004A	RW	CPU Speed Control
004B	RW	Command mode register
0061	RW	NMI Status
0070	RW	NMI Enable
0080-008F	RW	DMA Page registers
00A0	RW	INT 2 control register
00A1	RW	mask register
00C0-00DE	RW	DMA 2
00F0	W	80387 coprocessor Clear BUSY latch.
00F9	W	Clear Scratchpad RAM enable. (PRIVY)
00FB	W	Set Scratchpad RAM enable. (PRIVY)
0258	RW	EMS page 0 register.
0400-040F	RW	DMA 1
0461	RW	Extended NMI and reset control.
0462	RW	NMI I/O Interrupt port.
0464	RW	Last 32-bit bus master granted.
0480-048F	RW	DMA High page registers.
04C2	RW	reserved
04C6	RW	DMA 2 Channel 5
04CA	RW	DMA 2 Channel 6
04CE	RW	DMA 2 Channel 7
04D0	RW	INT 1 Edge level control register.
04D1	RW	INT 2 Edge level control register.
04D2-04DF	RW	reserved

PORT	RW	DESCRIPTION
04E0	RW	DMA Channel 0 Stop register bits <7: 2>
04E1	RW	DMA Channel 0 Stop register bits <15: 8>
04E2	RW	DMA Channel 0 Stop register bits <23: 16>
04E3	RW	reserved
04E4	RW	DMA Channel 1 Stop register bits <7: 2>
04E5	RW	DMA Channel 1 Stop register bits <15: 8>
04E6	RW	DMA Channel 1 Stop register bits <23: 16>
04E7	RW	reserved
04E8	RW	DMA Channel 2 Stop register bits <7: 2>
04E9	RW	DMA Channel 2 Stop register bits <15: 8>
04EA	RW	DMA Channel 2 Stop register bits <23: 16>
04EB	RW	reserved
04EC	RW	DMA Channel 3 Stop register bits <7: 2>
04ED	RW	DMA Channel 3 Stop register bits <15: 8>
04EE	RW	DMA Channel 3 Stop register bits <23: 16>
04EF	RW	reserved
04F0-04F3	RW	reserved
04F4	RW	First 1M mapping register.
04F4	RW	DMA Channel 5 Stop register bits <7: 2>
04F5	RW	DMA Channel 5 Stop register bits <15: 8>
04F6	RW	DMA Channel 5 Stop register bits <23: 16>
04F7	RW	reserved
04F8	RW	DMA Channel 6 Stop register bits <7: 2>
04F9	RW	DMA Channel 6 Stop register bits <15: 8>
04FA	RW	DMA Channel 6 Stop register bits <23: 16>
04FB	RW	reserved
04FC	RW	DMA Channel 7 Stop register bits <7: 2>
04FD	RW	DMA Channel 7 Stop register bits <15: 8>
04FE	RW	DMA Channel 8 Stop register bits <23: 16>
04FF	RW	reserved
08F4	RW	Bank 0 configuration register.
09F4	RW	Cache system status register.
0AF4	R	Cache TAG test register.
0BF4	RW	Cacheable memory control register.
0CF4	R	Cache identification register.
14F4	RW	First 1M configuration register.
18F4	RW	Bank 1 configuration register.
24F4	R	Parity error address register.
28F4	RW	Bank 2 configuration register.
34F4	RW	32-bit speed select register.
38F4	RW	Bank 3 configuration register.
4258	RW	EMS page 1 register. (Not used)
44F4	RW	16-bit speed select register.
48F4	RW	Bank 4 configuration register.
54F4	RW	8-bit speed select register.
58F4	RW	Bank 5 configuration register.
64F4	RW	Refresh/INTA speed select register.
68F4	RW	Bank 6 configuration register.
74F4	R	Bus/memory controller revision register.
78F4	RW	Bank 7 configuration register.
8258	RW	EMS page 2 register. (Not used)
C258	RW	EMS page 3 register. (Not used)

8. Power Requirements

The power requirement is yet to be determined, but should be similar to the existing Mongoose system and will in fact be powered by the same power supply.

Appendix A. PAL Equations

The following PAL equations are written in the *CUPL* device programming language. For a complete definition of the language see the *CUPL* user's manual. The following is a brief definition of the major operators.

Priority	CUPL syntax	Definition	Example
1	!	NOT	!A
2	&	AND	A & B
3	#	OR	A # B
4	\$	XOR	A \$ B

NOT		AND			OR			XOR		
A	!A	A	B	A&B	A	B	A#B	A	B	ASB
0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1	0	1	1
		1	0	0	1	0	1	1	0	1
		1	1	1	1	1	1	1	1	0

Signal Name Extensions

Extension	Description
.D	D Input of D-type flip-flop
.J	J Input of JK-type flip-flop
.K	K Input of JK-type flip-flop
.S	S Input of SR-type flip-flop
.R	R Input of SR-type flip-flop
.DQ	Q Output of D-type flip-flop
.AP	Asynchronous Preset of flip-flop
.AR	Asynchronous Reset of flip-flop
.SP	Asynchronous Preset of flip-flop
.SR	Asynchronous Reset of flip-flop
.CK	Programmable Clock of flip-flop
.OE	Programmable Output Enable
.CA	Complement Array

Appendix B. PAL Test Vectors

The following PAL test vectors were written in the *CUPL* device programming language. For information on it's syntax see appendix A.

Appendix B.1 Byte Enable Latch, 644-85

Appendix C. Reference Material

EISA Bus specification

This document is the established specification for the Enhanced Industry Standard Architecture bus.
Publisher: BCPR Services Incorporated

82357 (ISP) data sheet

Manufacturer's data sheet for the Integrated System Peripheral, which provides detailed hardware design information, and programming data.
Publisher: Intel Corporation

82358 (EBC) data sheet

Manufacturer's data sheet for the EISA bus controller, which provides detailed hardware design information.
Publisher: Intel Corporation

Advanced Work Station Technical Reference (595-3922)

This manual contains information on computer operation and programming methods, as well as a hardware introduction for the service technician.
Publisher: Zenith Data Systems

MS-DOS Programmer's Utility Pack (CB-3163-30)

This software packing produced by Microsoft and ZDS, contains information and software for writing and assembling MS-DOS and native code 8086, 80286, and 80386 assembly language programs.
Publisher: Zenith Data Systems

644-25 Posted Write Array

Internal ZDS document describing the hardware specification of the write queue gate arrays.
Publisher: Zenith Data Systems

644-28 High Performance 32/16-Bit Cache Memory Controller

Internal ZDS document describing the hardware specification of the system/cache controller gate array.
Publisher: Zenith Data Systems

644-29 32-Bit AT Bus/Memory Controller

Internal ZDS document describing the hardware specification of the bus/memory controller gate array.
Publisher: Zenith Data Systems

WTL3167 Floating Point Coprocessor

Manufacturer's data sheet which provides detailed information on hardware interfacing, electrical data, and programming.
Publisher: Weitek Corporation

Introduction to the 80386

This manual provides an overview and introduction to the 80386 microprocessor.
Publisher: Intel Corporation order number 231252

Microprocessor and Peripheral Handbook, Volumes 1 & 2

These manuals provide information on the architecture and programming procedures for the 80386 and 80387, and other support circuitry.

Publisher: Intel Corporation order number 230843-005.

80386 Programmer's Reference Manual

This manual contains detailed information on the architecture, instruction set, and programming procedures required for the 80386 microprocessor.

Publisher: Intel Corporation order number 230985-001

80386 Hardware Reference Manual

This manual contains detailed descriptions of the architecture and operation of the 80386 microprocessor. Device timing diagrams and pinouts are also included in this manual.

Publisher: Intel Corporation order number 231732-002.

80387 Programmer's Reference Manual

This manual contains detailed descriptions of the architecture, instruction set, and programming procedures required for the 80387 numeric coprocessor.

Publisher: Intel Corporation order number 231917-001.

LIM Expanded Memory Device Interface Specification

This specification details the software and hardware interface for EMS memory systems. Intel Technical Support provides a toll-free number 1-800-538-3373, that you may call to obtain this specification.

Publisher: Intel Corporation

Intel publications may be obtained by writing to the following address:

*Intel Literature Sales
P. O. Box 58130
Santa Clara, California 95052-8130*

or dial toll free: 1-800-548-4725

Weitek publications may be obtained by writing to the following address:

*WEITEK Corporation
1060 East Arques Avenue
Sunnyvale, California 94086-BRM-9759
ATTN: Ed Masuda*