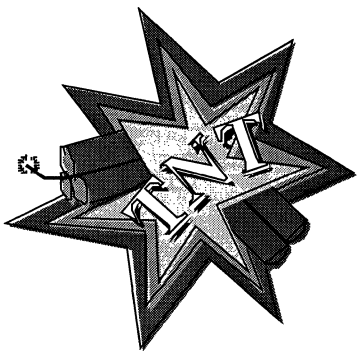
TNT Hammerhead ASIC ERS

Revision 1.1

September 25, 1995



Jim Kelly: 974-1062

Steve Polzin: 862-6497

QMServer: Design_Women

Apple Registered Confidential

This document represents the TNT Hammerhead ASIC Engineering Reference Specification. This document reflects Rev 6 Hammerhead silicon.

TABLE 1 Revision History

Who	When	Rev	What
J. Tringali	25-Feb-93	0.0	Limited Release
S.Polzin,J. Kelly	5-Apr-93	0.1	Filled in Cache, SDRAMs, 16272 stuff
S. Polzin,J.Kelly	13-Jan-94	0.2	Update to include Tsunami requirements and reflect real HH Rev 1 silicon. Chapter 2 and 4 still need work. Chapter 3 IS ACCURATE.
S.Polzin J.Kelly	19-Apr-94	1.0	Chapter 2 still needs work. Chapters 3 and 4 are accurate.
J.Kelly	25-Sep-95	1.1	Update to include the final recommended register programming values (see change bars in Chapter 3).

All versions following V1.0 will use change bars to highlight changes.

Table Of Contents

CHAPTER 1 Overview 5

1.1 Scope 5

1.2 Related Documents 5

1.3 Overview 7

1.4 Block Diagram Description 7

1.4.1 ARBus Arbiter 8

1.4.2 Memory Controller 9

1.4.3 L2 Cache Controller 10

CHAPTER 2 Functional Description 13

2.1 ARBus Arbiter 13

2.1.1 TNT ARBus Policy 14

2.1.2 System Arbitration and Priority 14

2.1.2.1 Address Bus Arbitration 14

2.1.2.2 Data Bus Arbitration 15

2.1.2.3 Address-Only Transactions 16

2.1.2.4 Address Bus Parking 16

2.1.3 Atomic Transactions 16

2.1.4 Deadlock Cases 16

2.1.4.1 IAmATurd 16

2.1.4.2 PCI to PCI 17

2.1.4.3 Need to Add 17

2.1.5 Time-outs 17

2.2 Memory Controller 17

2.2.1 Memory Performance 17

2.2.2 DRAM Timing Details 19

2.2.3 DRAM Refresh 20

2.3 L2 Cache Controller 21

2.3.1 L2 Cache Configurations 21

2.3.2 L2 Cache Initialization 22

2.3.3 L2 Cache Timing/Performance 22

2.3.4 L2 Cache Controller Operation 23

2.3.4.1 Cache Access Overview 23

2.3.4.2 General flow of events for a µprocessor read hit 23

2.3.4.3 General flow of events for a µprocessor read miss with allocate (no dirty victim) 24

2.3.4.4 General flow of events for a µprocessor read miss allocate with a dirty victim 25

2.3.4.5 General flow of events for a µprocessor write hit 27

2.3.4.6 General flow of events for a µprocessor write miss with allocate (no dirty victim) 28
2.3.4.7 General flow of events for a µprocessor write miss allocate with a dirty victim 28
2.3.5 Non-CPU accesses 32
2.4 ROM Access 33
2.5 Register Access 33

CHAPTER 3 Programmer's Model 35
3.1 Memory-Like Space 36
3.1.1 DRAM 36
3.1.2 ROM 36
3.2 System Control Registers 36
3.2.1 CPUID Register 40
3.2.2 Hammerhead Revision Register 41
3.2.3 Hammerhead MotherBoard ID Register 42
3.2.4 Hammerhead CPUSpeed Register 43
3.2.5 MotherBoard DRAM Configuration 44
3.2.6 Memory Timing Registers 45
3.2.7 Refresh Timing Register 46
3.2.8 ROM Timing Register 47
3.2.9 Arbiter Configuration Register 48
3.2.10 ARBus Timeout Register 49
3.2.11 Who Am I Register 50
3.2.12 Secondary Interrupt Register 51
3.2.13 Test Mode Register 52
3.2.14 L2 Cache Configuration Register 53
3.2.15 L2 Cache Diagnostic Register 55
3.2.16 L2 Cache TAG Registers 56
3.2.17 L2 Debug Register 57
3.2.18 DRAM Base Registers 58
3.2.18.1 Memory Sizing and Stitching Algorithm 60

CHAPTER 4 Interface Specification 61
4.1 Hammerhead Pins 61
4.2 Hammerhead Physical Pinout 64

CHAPTER 1

Overview

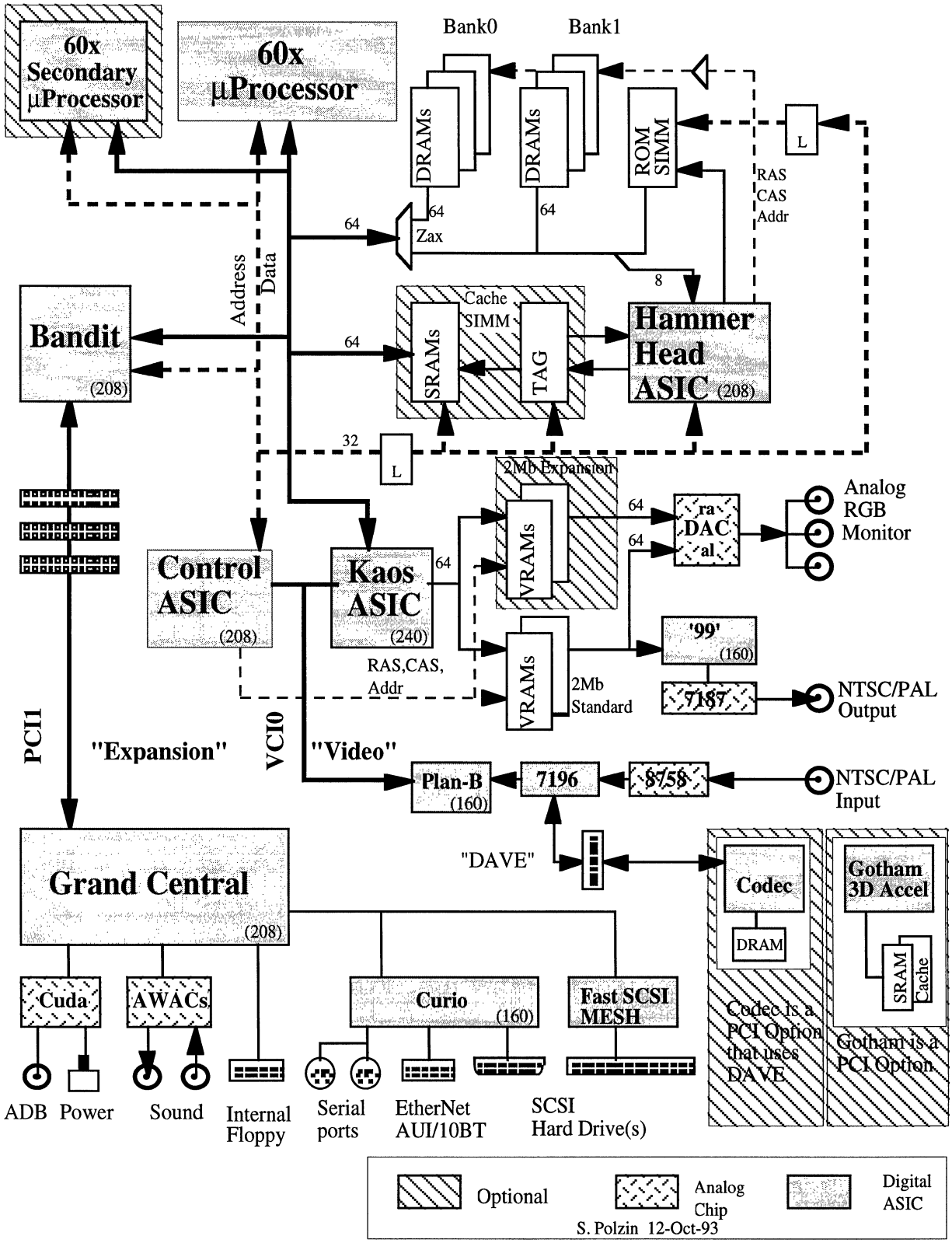
1.1 Scope

This document specifies the TNT Hammerhead ASIC. Hammerhead provides the ARBus arbiter, the DRAM controller, the Rom controller, and L2 Cache controller for the TNT family of systems (TNT, Nitro and Tsunami). Figure 1 is a block diagram of the TNT system.

1.2 Related Documents

- TNT System ERS: Steve Polzin et.al.
- TNT Control/Kaos ASIC ERS: Eric Baden et.al.
- TNT Bandit ASIC ERS: Steve Polzin et.al.
- Grand Central ASIC ERS: Kevin Christiansen et.al.
- djMEMC and MEMCjr ASIC ERS: Jay Rickard and Dale Adams.
- MacRISC Architecture Document: Ron Hochsprung ed.
- Apple RISC Bus: Steve Polzin ed.
- JEDEC Standard JC-42.5 for 8-Byte DRAM SIMMs.
- The 604 Microprocessor: PowerPC Implementation Definition and Functional Specification, IBM Corporation, V1.0 22 Apr 1992

FIGURE 1 TNT System Block Diagram



1.3 Overview

Hammerhead is designed to support flexible, high-performance memory systems for ARBus-based PowerPC Macintosh computers. It is designed to be logically and electrically compatible with the 601, 603 and 604 µprocessors and can operate at ARBus frequencies up to 50Mhz (150Mhz max µprocessor frequency). Hammerhead is packaged in a 208 PQFP and interfaces to the ARBus address lines and control pins, as well as the necessary DRAM, Rom, and L2 cache control pins. Hammerhead interfaces to the 8 most significant data bits in order to support internal register accesses.

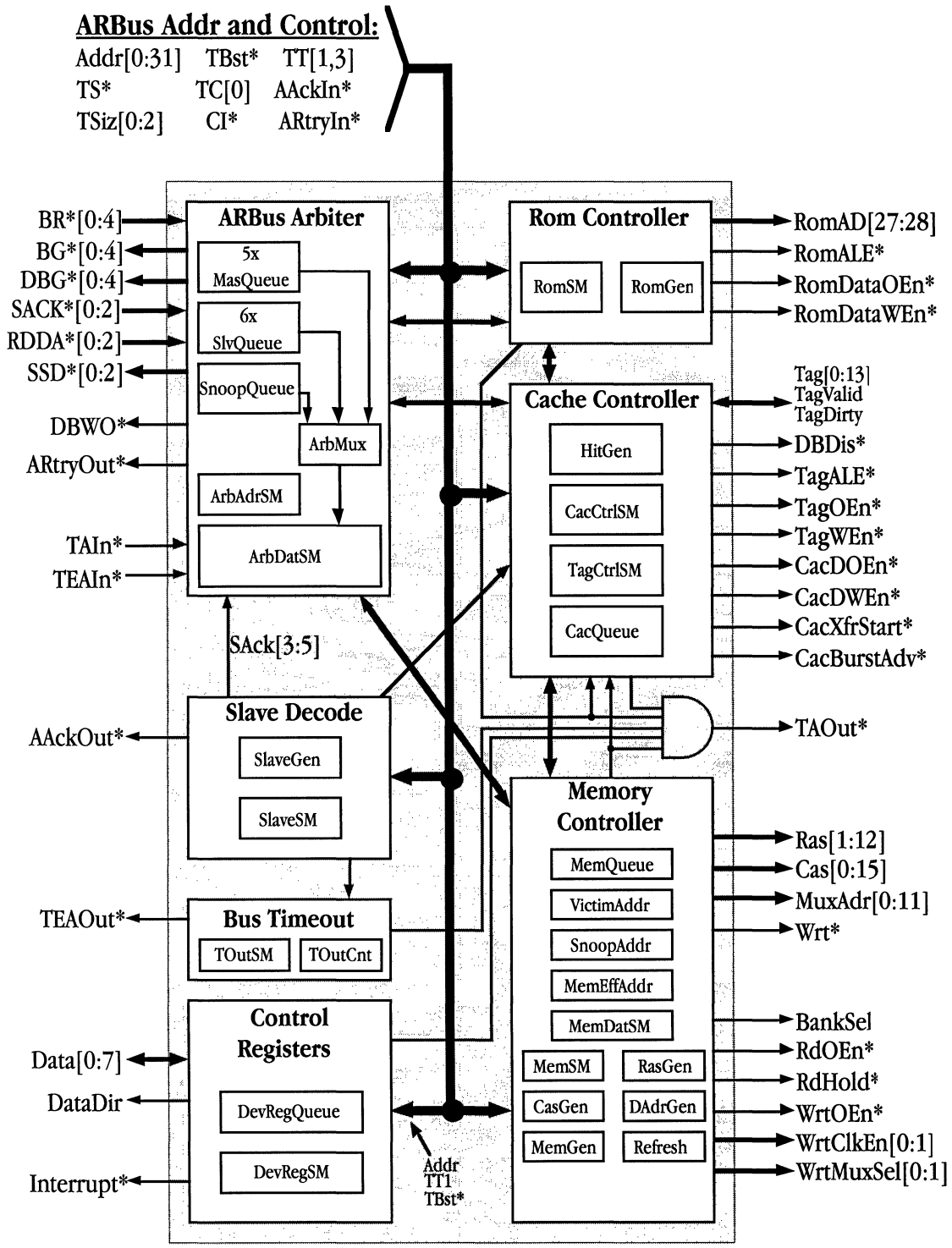
Hammerhead can operate in a number of modes that are selectable either at reset (via pullup/downs) or via software controllable registers. The following lists the modes in which Hammerhead can operate.

- DRAM System
  - 128-bit wide (2 bank) DRAM with single 64-bit SIMM upgrades. This allows single SIMM upgrades when used with 64-bit JEDEC SIMMs
  - Large memory support for 12 64-bit SIMMs
  - Optimum performance for 60ns DRAMs with ARBus frequencies between 33MHz and 50MHz.
- L2 Cache
  - 256KB, 512KB, 1MB, or 4MB size
  - Write-Back
  - 3:1:1:1 with 601
  - 2:1:1:1 with 603/604 "Fast L2 Mode"
- Arbiter
  - DBG controlled 601 style
  - DBG and DBDIS controlled 603/604 style
- ROM
  - Optimum performance for ARBus frequencies between 33MHz and 50MHz

1.4 Block Diagram Description

Figure 2 is a block diagram of the Hammerhead ASIC and its relation to the storage elements it controls.

FIGURE 2 Hammerhead ASIC Block Diagram



1.4.1 ARBus Arbiter

The ARBus Arbiter is designed to support 601, 603 and 604 PowerPC µprocessors. The 601 is the first PowerPC µprocessor and behaves according to the original 601 bus specification. The 603 and 604 are very similar and have added features that allow for a faster L2 cache (DBDis signal) and have a more elegant solution to certain write-queue conditions (IAmATurd?). Hammerhead is configured by software to recognize the µprocessor type (see Section 3.2.6 on page 45). The Hammerhead ARBus arbiter allows for a split transaction protocol whereby the address and data tenure ordering is not strictly enforced. The basic behavior that the arbiter guarantees is:

- Any given ARBus master has its own address and data tenures strictly ordered. This implies that DBG(n) always corresponds to TS(n) and for a set of TS(n) and TS(n+1), DBG(n) will always occur before DBG(n+1).
- Any given ARBus slave has its own data tenures strictly ordered. This implies that SSD(n) always corresponds to TS(n) and for a set of TS(n) and TS(n+1), SSD(n) will always occur before SSD(n+1).
- Data bus tenure is not necessarily granted on the ARBus in the same order as address tenure is granted if the address tenures are granted to different masters. This implies that if TS(n) is for Master A and TS(n+1) is for Master B, DBG(n) *may* be for Master B and therefore DBG(n+1) for Master A.

The Hammerhead ARBus arbiter is designed to support a single 60x µprocessor. A spare set of BR,BG,DBG pins are allocated (and given lowest priority) for future asymmetric multi-processing (603 accelerators etc.). The arbiter supports 5 logical masters (Video, Two Bandits, Primary 60x, Slave 60x) and 6 logical Slaves (Memory, Rom, Hammerhead registers, Video, and Two Bandits). The five masters arbitrate for use of the bus at a fixed priority as follows:

1. Highest: Video Bridge (Kaos/Control)
2. Bandit 1 (Grand Central lives here)
3. Bandit 2
4. Primary µprocessor
5. Lowest: Slave µprocessor

ARTRY is used by the primary µprocessor to signal dirty L1 cache block conditions that are resolved with subsequent castouts. All other nodes assert ARTRY ONLY to avoid deadlocks (see Section 2.1.4 on page 16). In general, AACK is delayed to enforce flow control. When the L2 Cache is installed, writeback cache victim transfers use the ARBus wires but do not strictly follow the ARBus protocol (see Section 2.3.4 on page 23).

1.4.2 Memory Controller

The Hammerhead Memory Controller supports a 128-bit wide 2-bank interleaved operation or 64-bit wide single bank operation selectable on a per-bank basis (see Section 3.2.18 on page 58). When both banks in a pair are present, both banks are cycled at the same time, with data from one 64-bit bank data transferred on the ARBus before the data from the other 64-bit bank (which bank goes first is a function of the ARBus critical word first coding of ARBusAddr[28:31]). This implies that for an ARBus 32-byte burst transaction, the DRAMs are cycled two times. The Zax ASIC selects between the two banks. When a single bank in a pair is present, the bank is cycled four times.

Using 4M-bit technology DRAMs, the baseline (soldered on the motherboard) memory is a single interleave set (two banks) yielding 8M-bytes (sixteen 512Kx8). Alternatively, thirty-two (32) 1Mx4 parts could be soldered to the motherboard as a single interleave set yielding 16M-bytes of on-board memory. Table 2 lists the maximum memory sizes supported by the TNT family.

TABLE 2 Maximum Memory Sizes Supported

Dram Density	SIMM Configuration	SIMM Capacity	Maximum Memory with 8 SIMMs (TNT)	Maximum Memory with 12 SIMMs (Tsunami)
4 MBit	256K x 64	2MBytes	16MBytes	24MBytes
4 MBit	512K x 64	4MBytes	32MBytes	48MBytes
4 MBit	1M x 64	8MBytes	64MBytes	96MBytes
16MBit	1Mx64	8MBytes	64MBytes	96MBytes
16MBit	2Mx64	16MBytes	128MBytes	192MBytes
16MBit	4Mx64	32MBytes	256MBytes	384MBytes
64MBit	4Mx64	32MBytes	256MBytes	384MBytes
64MBit	8Mx64	64MBytes	512MBytes	768MBytes
64MBit	16Mx64	128MBytes	1GBytes	1.5GBytes

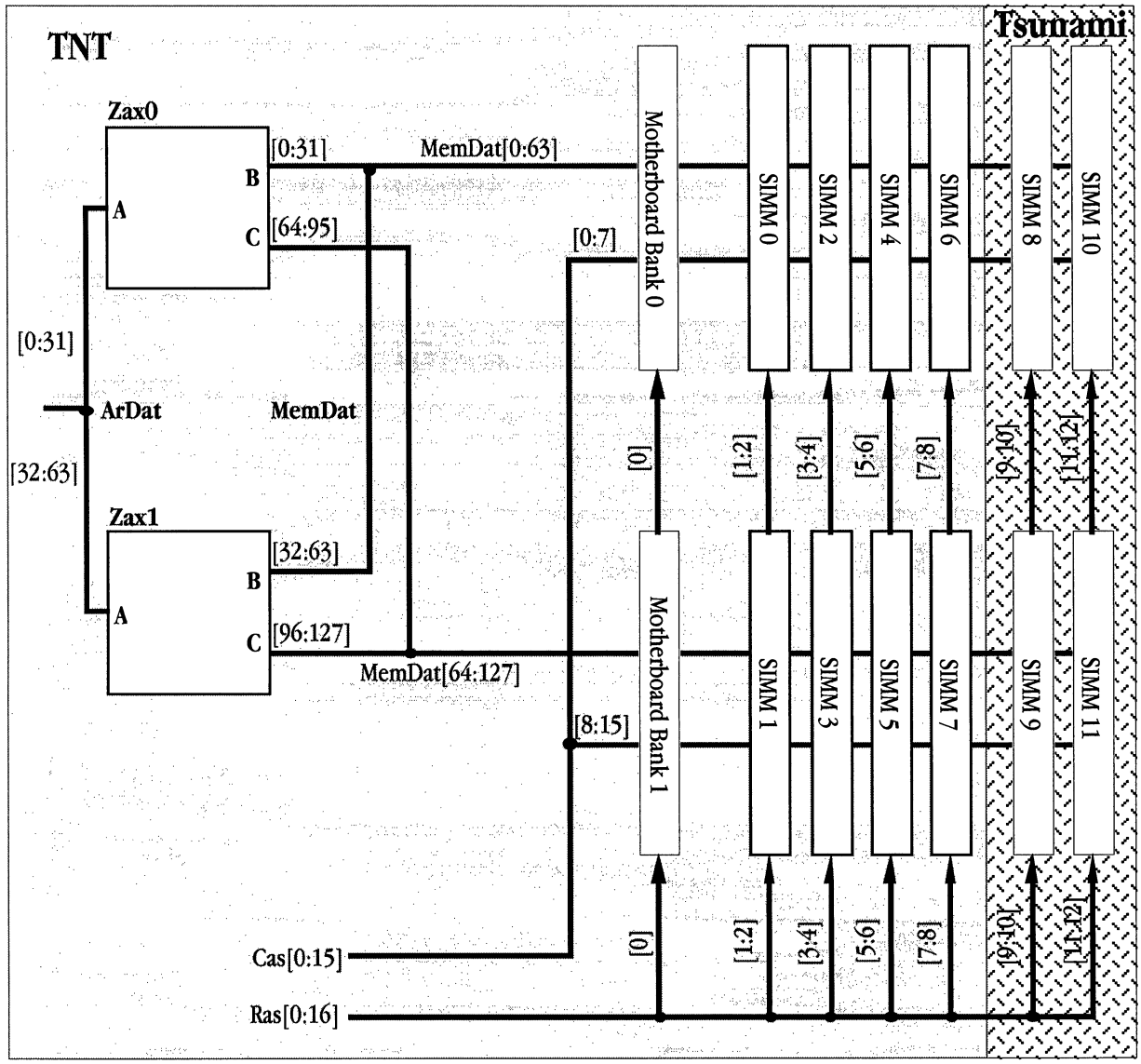
The Hammerhead Memory Controller provides a total of 13 RAS lines, configured as shown in Figure 3.

1.4.3 L2 Cache Controller

The Hammerhead L2 Cache controller implements a direct-mapped, write-back, allocate-on-burst cache. Cache is coherent with memory. Cache obeys the CI wire protocol. Transfers from the PCI busses can elect to not use the L2 via the proper assertion of this wire (selectable via a register in Bandit). The L2 cache is mode-bit selectable I, D, or I&D stream, on/off (see Section 3.2.14 on page 53). The Cache Controller makes no attempt to keep the L1 cache in the primary µprocessor a proper subset of L2 (inclusion). The L2 cache size is programmable at reset (via WrtClkEn[0:1] (see Figure 27 on page 53) from 256K to 4Meg. The TNT cache uses industry-standard synchronous burst-mode SRAMs (same as those used on 486 and Pentium machines) for the data store and generic SRAMs for the TAG store.

The Hammerhead Cache Controller also controls the response to ROM accesses. Up to 16M-bytes of ROM are supported. Hammerhead also supports writing the ROM for use with FLASH ROMs. ROM access time is programmable (see Section 3.2.6 on page 45). The ROM is read with 8-byte or 32-byte (burst) transactions from the ARBus. ROM space is writable with 8-byte transactions (no byte masking) for FLASH support.

FIGURE 3 Hammerhead Memory RAS/CAS wiring



Note: On TNT Protos 1 and 2, SIMMs 0 to 3 shown above are actually composed of two 32-bit SIMMs each.

Hammerhead Bank Numbering Scheme:

Bank 0: Motherboard Bank 0	Bank 8: SIMM 2 Back	Bank 16: SIMM 6 Back	Bank 24: SIMM 10 Back
Bank 1: Motherboard Bank 1	Bank 9: SIMM 3 Back	Bank 17: SIMM 7 Back	Bank 25: SIMM 11 Back
Bank 2: SIMM 0 Front	Bank 10: SIMM 4 Front	Bank 18: SIMM 8 Front	
Bank 3: SIMM 1 Front	Bank 11: SIMM 5 Front	Bank 19: SIMM 9 Front	
Bank 4: SIMM 0 Back	Bank 12: SIMM 4 Back	Bank 20: SIMM 8 Back	
Bank 5: SIMM 1 Back	Bank 13: SIMM 5 Back	Bank 21: SIMM 9 Back	
Bank 6: SIMM 2 Front	Bank 14: SIMM 6 Front	Bank 22: SIMM 10 Front	
Bank 7: SIMM 3 Front	Bank 15: SIMM 7 Front	Bank 23: SIMM 11 Front	

JDK, 6Apr94

CHAPTER 2

Functional Description

This chapter describes the operational details of each major portion of the Hammerhead ASIC.

2.1 ARBus Arbiter

The Hammerhead ARBus Arbiter is designed to support 601,603 and 604 PowerPC µprocessors. The 601 is the first PowerPC µprocessor and behaves according to the original 601 bus specification. The 603 and 604 are very similar and have added features that allow for a faster L2 cache (DBDis signal) and have a more elegant solution to certain write-queue conditions (IAmATurd?). Hammerhead is configured via ArbConfig.ModPPC (see Figure 22 on page 48) to set the µprocessor type (processor variations only affect cache behavior). The Hammerhead ARBus arbiter allows for a split transaction protocol whereby the address and data tenure ordering is not strictly enforced. The basic behavior that the arbiter guarantees is:

- Any given ARBus master has its own address and data tenures strictly ordered. This implies that DBG(n) always corresponds to TS(n) and for a set of TS(n) and TS(n+1), DBG(n) will always occur before DBG(n+1).
- Any given ARBus slave has its own data tenures strictly ordered. This implies that SSD(n) always corresponds to TS(n) and for a set of TS(n) and TS(n+1), SSD(n) will always occur before SSD(n+1).
- Data bus tenure is not necessarily granted on the ARBus in the same order as address tenure is granted if the address tenures are granted to different masters. This implies that of TS(n) is for Master A and TS(n+1) is for Master B, DBG(n) *may* be for Master B and therefore DBG(n+1) for Master A.

The Hammerhead ARBus arbiter is designed to support a single 60x µprocessor. A spare set of DBG,BG,BR pins are allocated (and given lowest priority) for future asymmetric multi-processing (603 accelerators etc.)

ARTRY is used by the primary µprocessor to signal dirty L1 cache block conditions that are resolved with subsequent castouts. All other nodes assert ARTRY ONLY to avoid deadlocks (see Section 2.1.4 on page 16). In general, AACK is delayed to enforce flow control. When the L2 Cache is installed, writeback cache victim transfers use the ARBus wires but do not strictly follow the ARBus protocol (see Section 2.3 on page 21).

2.1.1 TNT ARBus Policy

The Hammerhead ARBus Arbiter provides the following ARBus policy to the 60x µprocessor:

- Flow control is accomplished by delaying the assertion of AACK. If the node selected by a TS/ Address cycle generated by the 60x cannot service the access at the time, AACK is delayed until the node is able to service the access.
- 60x DBB and ABB are not connected to any logic in the TNT system.
- 60x DBWO is wired permanently de-asserted.
- L1 cache coherence support for global references is supported using the 60x ARTRY protocol.

2.1.2 System Arbitration and Priority

The Hammerhead Arbiter supports five logical masters (Control/Kaos VCI, two Bandit Expansion PCIs, Primary 60x, Slave 60x) and six Logical Slaves (Control/Kaos VCI, two Bandit Expansion PCIs, Memory (Hammerhead), Rom, and Hammerhead registers). The five masters arbitrate for use of the bus at a fixed priority as follows:

1. Highest: Control/Kaos
2. Expansion Bandit 1 (Grand Central lives here)
3. Expansion Bandit 2
4. Primary µprocessor
5. Lowest: Slave µprocessor

The ARBus Arbiter may optionally (see Figure 22 on page 48) 'park' the 60x processor or the video Bandit on the ARBus by asserting the appropriate BG wire during idle bus cycles. The default mode of operation is to park the most recent master.

By giving highest priority to Control/Kaos, Hammerhead allows a PCI Master (Plan-B Video In) to "hog" the ARBus.

2.1.2.1 Address Bus Arbitration

The arbiter controls five masters (Control/Kaos, two Expansion Bandits, primary µprocessor and slave µprocessor) and six slaves (Control/Kaos, two Expansion Bandits, Memory, Rom, and Hammerhead registers).

Address bus arbitration occurs in every cycle that an address tenure is not active. Masters assert their individual bus request signals (BR) to the arbiter to signal a request for service. The arbiter signals the node which won the arbitration by asserting bus grant (BG). Nodes that have BG asserted in a given cycle are free to assert TS and therefore start a transaction in the next cycle. In the cycle immediately following the assertion of ARTRY by the 60x µprocessor, the arbiter does not allow nodes other than the 60x to win use of the address bus. Figure 4 documents the Address Bus Arbitration Algorithm.

FIGURE 4 Address Bus Arbitration Algorithm

```
function [0:4] CalcBG;

      input [0:4] BR;
      input [0:4] OldBG;
      input       ParkMode;
      input [0:2] ParkVal;

      if (&BR) begin
            if (!ParkMode) CalcBG = OldBG;
            else case (ParkVal)           // synopsys full_case parallel_case
                  2'b000: CalcBG = 5'b01111;    // Control/Kaos
                  2'b001: CalcBG = 5'b10111;    // Bandit 1 (GC/Expansion)
                  2'b010: CalcBG = 5'b11011;    // Bandit 2
                  2'b011: CalcBG = 5'b11101;    // CPU0
                  2'b100: CalcBG = 5'b11110;    // CPU1
            endcase
      end else casex (BR)                 // synopsys full_case parallel_case
            5'b0xxxx: CalcBG = 5'b01111;        // Control/Kaos
            5'b10xxx: CalcBG = 5'b10111;        // Bandit 1 (GC/expansion)
            5'b110xx: CalcBG = 5'b11011;        // Bandit 2
            5'b1110x: CalcBG = 5'b11101;        // CPU0
            5'b11110: CalcBG = 5'b11110;        // CPU1
      endcase

endfunction
```

2.1.2.2 Data Bus Arbitration

The arbiter controls the use of the data signals as a function of the address and the availability of read data. The arbiter decodes all ARBus addresses and generates an implied Slave Acknowledge. If a given ARBus address receives an AACK, by decoding the address, the arbiter knows which slave will accept write data or will return read data. A slave that asserts AACK for a write transaction gives implicit permission to the arbiter to grant the data bus to the master and allow it to assert TA and the associated write data. Slaves must assert RDDA when requested return read data is available.

The arbiter grants the data bus to the selected master via the assertion of DBG (Data Bus Grant) and indicates to the slave that data is to be asserted or accepted via the assertion of SSD (Source or Sync Data). Figure 4 documents the Data Bus Arbitration Algorithm.

FIGURE 5 Data Bus Arbitration Algorithm

```
(a pseudo-code summary is more appropriate here)

      •if at least one master queue is non-empty
            •select the highest priority non-empty master queue,
             based upon the following priority encoding:
                  0:    Kaos/Control      (Highest)
                  1:    Bandit 1
                  1:    Bandit 2
                  2:    CPU 0
```

```
            3:    CPU 1                (Lowest)
          •upon examining the selected master queue to see which
           slave is selected in the front entry, look at the front
           entry of the associated slave queue to see if it
           points back to the selected master. If it does,
           a master/slave match occurs.
          •if a master/slave match has occurred, grant the data bus
           to the selected master (via DBG) and slave (via SSD).
           Otherwise, remain idle.
    •otherwise remain idle
```

2.1.2.3 Address-Only Transactions

Transactions which do not involve a data transfer (Address-Only transactions) are only generated by the 60x µprocessor and are simply acknowledged (AACK asserted) by the arbiter with no other nodes participating.

2.1.2.4 Address Bus Parking

When the address bus is idle, the Arbiter asserts BG to the master specified by the Parked Master Register (see Figure 22 on page 48) until the Arbiter receives a new BR. This "parked" master can immediately assert TS to begin a transaction without the two cycles of delay (BR cycle and BG cycle) associated with address arbitration. When a new master requests the address bus, the Arbiter de-asserts the parked master's BG signal and waits one cycle to assert BG to the new master to avoid conflicts with the parked master. The cycle-by-cycle flow is as follows:

1. Arbiter 'parks' a given master (the parked master) by asserting BG.
2. A new master requests use of the bus by asserting its BR.
3. The arbiter de-asserts BG to the parked master. If TS is not asserted in this cycle, the arbiter will assert BG in the next cycle.
4. If TS was not asserted in step 3, the arbiter asserts BG to the new master in this cycle.

2.1.3 Atomic Transactions

The 60x µprocessor uses LARX and STCX instruction sequences for performing atomic accesses (read-modify-write) to memory. PCI nodes use the PCI LOCK# wire/protocol to perform atomic accesses to memory. In neither case is Hammerhead involved. All atomic access facilities in TNT are handled elsewhere (see TNT System ERS V1.0)

2.1.4 Deadlock Cases

Hammerhead and Bandit work together to prevent any deadlock scenarios from happening as a result of interfacing more than one single envelope (non-split-transaction) busses (PCI0, PCI1) to a split transaction bus (ARBus). Two scenarios are of interest.

2.1.4.1 IAmATurd

TNT systems with the 601 have a potential for "Write Deadlock." Deadlock will occur if the 601 executes a series of writes to PCIs and fills the 601's internal write queue, while a PCI Master executes a read of memory that hits dirty in the cache of the 601. At this point the PCI is owned by the master performing the read of memory and cannot service the pending writes from the 601. The PCI read of memory cannot complete until the dirty data in the 601's cache is pushed to memory. The 601 does not re-order the snoop push around the

writes already pending in its write queue. Since the read cannot complete until the writes complete and the writes cannot complete until the read completes, the system is deadlocked.

The 601 provides a pin called IAmATurd which forces it to always re-route snoop pushes around pending writes in the write queue. 603 and 604 µprocessors behave in this manner by default. TNT asserts the 601 IAmaTurd pin at all times to force the behavior.

2.1.4.2 PCI to PCI

When a master on one PCI reads an address that maps to the other PCI while a similar operation occurs in the opposite direction, a "Passing in the night" deadlock scenario is possible. TNT avoids this by propagating ARTRY from the ARBus to the PCI. The behavior in Bandit is as follows:

1. If a PCI read gets retried on the ARBus, it also gets retried on the PCI via target disconnect.
2. Bandit ARTRYs an ARBus read transaction that references an address on its PCI if a read from a master on its PCI is pending.
3. If an ARBus read gets retried on the PCI, ARBus is NOT retried. Once an ARBus read is accepted by the bridge (i.e. there is a buffer available), is not retried on the ARBus, its is only retried on the PCI.

2.1.4.3 Need to Add

1. 3 Transaction limit
2. 2 Transaction limit to PCI
3. If a master has a PCI read outstanding, artry any non-PCI transactions from that same master.

2.1.5 Time-outs

The Hammerhead ARBus Arbiter enforces a limit in the amount of time a slave can delay asserting AACK to prevent system hangs and allow software to size the system at start-up. The ARBus time-out register (see Figure 24 on page 50) controls the time interval that the Arbiter allows between TS and AACK. If the time-out counter expires, the Arbiter asserts AACK followed by TA along with TEA to signal to the requesting master that something has failed. This sequence results in a machine check exception being generated to the processor. All Slaves are required to cancel the given transaction gracefully when this event occurs.

2.2 Memory Controller

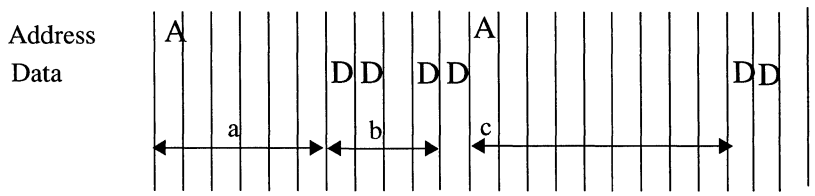
Main memory control is provided through Hammerhead. The port to memory is control only, i.e., there is no memory datapath through the Hammerhead ASIC. Hammerhead is designed to support motherboard-resident and user-upgraded memory arrays. The Hammerhead Memory Controller supports either 128-bit wide 2-bank interleaved operation or 64-bit wide single bank operation.

2.2.1 Memory Performance

The Hammerhead memory controller is synchronous to the system transfer clock. For the TNT family the maximum transfer clock frequency is 50MHz. The latency to the first 64-bit transfer on any memory reference is targeted to be no more than 6 system transfer clocks exclusive of any system arbitration overhead. During processor block operations the following three 64-bit transfers are returned in four clocks. Independent back-to-back operations require an additional 3 clocks of processing overhead for RAS pre-charge. For system level

performance data the overhead of the arbiter and Processor bus controller must be taken into account. Figure 6 illustrates the basic 128-Bit DRAM Memory pipeline. Figure 6 illustrates the basic 64-bit DRAM pipeline.

FIGURE 6 128-Bit DRAM Memory Pipeline (32-byte aligned)

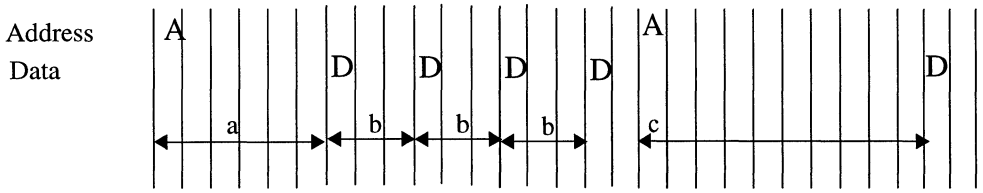


128 bit wide memory subsystem

TABLE 3. DRAM Memory Pipeline Parameters for 50Mhz ARBus and 60ns DRAMs

Parameter	System Clocks	Symbol	MemTime Register Values
First Doubleword Latency - Idle	6	a	0b010 (See Table 28 on page 53)
CAS Cycle Time Rate (MemTime.RdCasTime)	3	b	0b10 (See Table 28 on page 53)
First Doubleword Latency - Busy	10	c	0b110 (See Table 28 on page 53)

FIGURE 7 64-Bit DRAM Memory Pipeline



64 bit wide memory subsystem

Hammerhead supports the 601 style critical quad-word first and 603/604 style critical double-word first data ordering as a function of ARBus Address [27:28] as shown in Table 4 and Table 4.

TABLE 4 128-bit Critical Word First Behavior (assuming parameters in Table 3)

Data Cycle	Addr[27:28] = 00	Addr[27:28] = 01	Addr[27:28] = 10	Addr[27:28] = 11
First	Double-Word 0 in Cycle 6	Double-Word 1 in Cycle 6	Double-Word 2 in Cycle 6	Double-Word 3 in Cycle 6
Second	Double-Word 1 in Cycle 7	Double-Word 2 in Cycle 8	Double-Word 3 in Cycle 7	Double-Word 0 in Cycle 8
Third	Double-Word 2 in Cycle 9	Double-Word 3 in Cycle 9	Double-Word 0 in Cycle 9	Double-Word 1 in Cycle 9
Fourth	Double-Word 3 in Cycle10	Double-Word 0 in Cycle11	Double-Word 1 in Cycle10	Double-Word 2 in Cycle11

TABLE 5 64-bit Critical Word First Behavior (assuming parameters in Table 3)

Data Cycle	Addr[27:28] = 00	Addr[27:28] = 01	Addr[27:28] = 10	Addr[27:28] = 11
First	Double-Word 0 in Cycle 6	Double-Word 1 in Cycle 6	Double-Word 2 in Cycle 6	Double-Word 3 in Cycle 6
Second	Double-Word 1 in Cycle 9	Double-Word 2 in Cycle 9	Double-Word 3 in Cycle 9	Double-Word 0 in Cycle 9
Third	Double-Word 2 in Cycle 12	Double-Word 3 in Cycle 12	Double-Word 0 in Cycle 12	Double-Word 1 in Cycle 12
Fourth	Double-Word 3 in Cycle15	Double-Word 0 in Cycle15	Double-Word 1 in Cycle15	Double-Word 2 in Cycle15

2.2.2 DRAM Timing Details

Table 6 shows the timing budget for the DRAM-based TNT Memory system. Initial TNT systems use 60ns DRAMs to provide optimum performance. Via the Memory Timing registers (see Figure 18 on page 45), Hammerhead can be programmed to support slower DRAMs in future low-end systems.

TABLE 6 DRAM 1st Word Timing Details

Timing Component	Delay	Comments
60x Clock to Output	10ns	601 spec
Tprop ArAddr	2ns	
Setup to HH Flop	2ns	TI Spec

TABLE 6 DRAM 1st Word Timing Details

Timing Component	Delay	Comments
HH Clock Skew	1ns	
1st Cycle Total	15ns	
Begin Async Delay		
HH Mux Logic	2ns	
HH Output Latch	5ns	
HH Driver	4ns	
1004 Buffer	5ns	
Tprop BuffAddr	5ns	
Tras	60ns	
Tprop MData	5ns	
Tsu 16272	2ns	
Clock Skew	5ns	Buffered SysClk @Zax
Total Async	88ns	round up to 5 cycles @50MHz
Begin ARBus Data Transfer		
Tckout 16272	5ns	
Tprop ArData	5ns	
Tsu 601	2ns	
Clock Skew	5ns	Buffered SysClk @Zax
Total TA Cycle	17ns	
Total to First Word		7 cycles @50MHz

Note: Preliminary Timing! Danger Will Robinson!

2.2.3 DRAM Refresh

Hammerhead's refresh strategy employs a promoted priority scheme. Refresh counter overflows increment a small counter that keeps track of pending refresh requests. If the number of pending refresh requests is three or less, then a low-priority refresh request is issued to the memory controller. Such a low-priority request is serviced by the memory controller only if no external read/write requests are queued.

If the system is in the middle of a large block transfer, it is possible for refresh requests to be "choked". If this condition persists, memory decay will eventually result. For this reason, the refresh request will switch from low to high priority as soon as four (or more) refresh requests are pending. Such a high-priority refresh request is serviced by the memory controller before all other requests. This strategy guarantees forward

progress for refreshes, while keeping latency due to refresh low in most cases. Even so, the worst-case system latency adder due to refresh is less than 320nS.

This refresh strategy also makes it exceedingly simple to perform the correct DRAM wake-up procedure upon system power-up. This procedure requires an initial idle time of at least 100 microseconds (accomplished during the reset interval), followed by eight refresh cycles. Initializing the pending refresh counter to 11 (in hardware) achieves the desired behavior.

There exists a duality of refresh modes for the 16Mb DRAM devices in the JEDEC world. In order to maintain the highest degree of configuration flexibility, Hammerhead only supports the 11x11, or 2048, refresh cycle devices for the 16Mb technology DRAMs.

2.3 L2 Cache Controller

The Hammerhead Cache Controller (CC) contains the necessary logic to control the write-back L2 cache. The L2 cache is organized as a write-back, direct mapped (single set) cache with allocate on read or write. The cache data store is implemented with synchronous static RAMs with the TAG store implemented with 'standard' SRAMs. The size of the second level cache is programmable from 256K to 4Meg during reset via WrtClkEn[0:1] (see Figure 27 on page 53).

2.3.1 L2 Cache Configurations

The L2 cache is implemented on a SIMM that uses synchronous burst SRAMs from Motorola or Micron for the data store and a standard SRAM from Motorola or Micron for the TAG/Status store. Table 7 shows the cache size options and TAG/Index division. The second level cache is direct-mapped and is write-back. The Bus Controller can be configured to only allow processor I-Stream references to be cached, only D-Stream references to be cached, or both (see Figure 17 on page 44 for a description of the cache control options). The Cache Controller also follows the ARBus CI (Cache-Inhibit) wire. Table 10 on page 31 shows the detailed behavior of the cache and memory system given ARBus transactions. Address-only ARBus transactions have no affect on the state of a given line in the second level cache.

TABLE 7 Second Level Cache Size Options

Cache Size	Tag/Status Store Standard SRAM	Data Store Synch Burst SRAM	TAG	Index	Wrap
256 K-Bytes	One 8Kx16	Four 32Kx16 or Eight 32Kx8	Addr[0:13]	Addr[14:26]	Addr[27:28]
512 K-Bytes	***One 16Kx16***	***Four 64Kx16 or Eight 64Kx8***	***Addr[0:12]***	***Addr[13:26]***	***Addr[27:28]***
1 M-Bytes	One 32Kx16	Four 128Kx16 or Eight 128Kx8	Addr[0:11]	Addr[12:26]	Addr[27:28]
4M-Bytes	One 128Kx16 or Two 32Kx16	Four 512Kx16 or Eight 512Kx8	Addr[0:9]	Addr[10:26]	Addr[27:28]

2.3.2 L2 Cache Initialization

The TNT L2 Cache uses a standard SRAM for the TAG and status store which needs to be initialized before the cache is enabled. Initialization or diagnostic software can initialize the Valid and Dirty bits via the L2Diag register (see Figure 17 on page 44). When L2Diag.Ena is set, any access to memory space forces the Cache Controller to write the value of L2Diag.Dirty and L2Diag.Valid into the TAG Store RAM. To invalidate the Cache, software needs to set L2Diag.Ena, clear L2Diag.Valid and loop through reads of the first `cache_size` number of bytes in memory.

2.3.3 L2 Cache Timing/Performance

The L2 cache is designed to work with 60x-based systems that have µprocessor frequencies up to 150MHz and ARBus frequencies up to 50MHz. Using 11ns synchronous burst SRAMs, the first word of data is presented to the processor in the ARBus cycle immediately following the assertion of ARBus TS for 'hits'. The timing budget for this access is shown in Table 8 and Table 9.

Note: Real early Data
The following tables contain early data and will be updated when firmer clock skew and bus propagation numbers are available

TABLE 8 L2 Cache Timing Budget (First Cycle)

Timing Component	Delay	Comments
60x Clock to Output	10ns	601 spec
Tprop Address	2ns	
Address Buffer	4ns	
Setup to SRAM	2ns	Moto Spec
Clock Skew	1ns	
1st cycle Total	19ns	

TABLE 9 L2 Cache Timing Budget (Second Cycle)

Timing Component	Delay	Comments
SRAM Clock to Output	11ns	Moto Spec
Tprop Data	2ns	
Setup to 60x	2ns	601 spec
Clock Skew	4ns	
2nd cycle Total	19ns	

2.3.4 L2 Cache Controller Operation

The second level cache allocates on reads from DRAM addresses responded to by the Memory Controller and ROM addresses. The cache only allocates on burst read and write misses. Non-Burst reads that hit in the cache are serviced from the cache. Non-burst writes that hit in the cache cause the given line to be written back to memory before the single cycle of write data is written to memory. A given address/TS is defined to be a hit in the L2 cache if the TAG at the given index matches AND the Valid bit is set.

2.3.4.1 Cache Access Overview

The Cache Controller is able to assert data (ARBus TA) for reads that hit in the L2 cache by taking advantage of some other system attributes:

1. With only a single snooping node (one processor), all µprocessor accesses that 'hit' in the L2 cache never have ARTRY asserted.
2. By using the newly defined DBDIS (Data Bus Disable) wire for both the 603 and 604, the Cache Controller asserts DBG in the same cycle that TS asserts, allowing the processor to accept data (DBB) in the very next cycle. DBDIS allows the L2 cache RAMs to be enabled, driving the data wires before the direction of the access (read or write) is determined.
3. DRTRY is never asserted, allowing the use of the non-aging mode in the 603 and 604.

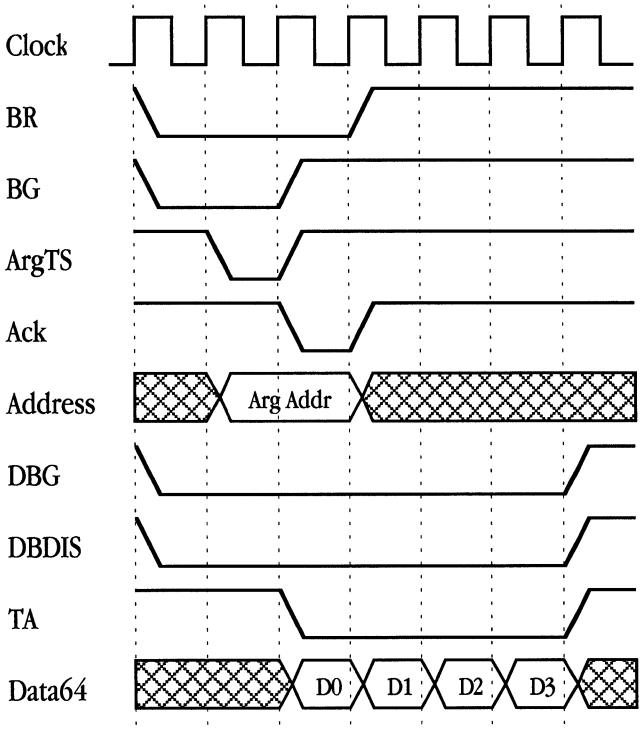
For 601-based systems, the earliest assertion of TA is in cycle 2 (counting TS assertion to be in cycle 0).

2.3.4.2 General flow of events for a µprocessor read hit

See Figure 8.

1. TS starts lookup, DRAM cycle is started (RAS assertion etc.). Memory Controller asserts AACK.
2. TAG declares a hit. The Cache Controller asserts L2HIT to Memory Controller.
3. DRAM cycle is aborted gracefully.
4. Data is sourced from the Cache RAMs. The Cache Controller asserts TA appropriately.

FIGURE 8 Read Hit Timing Diagram

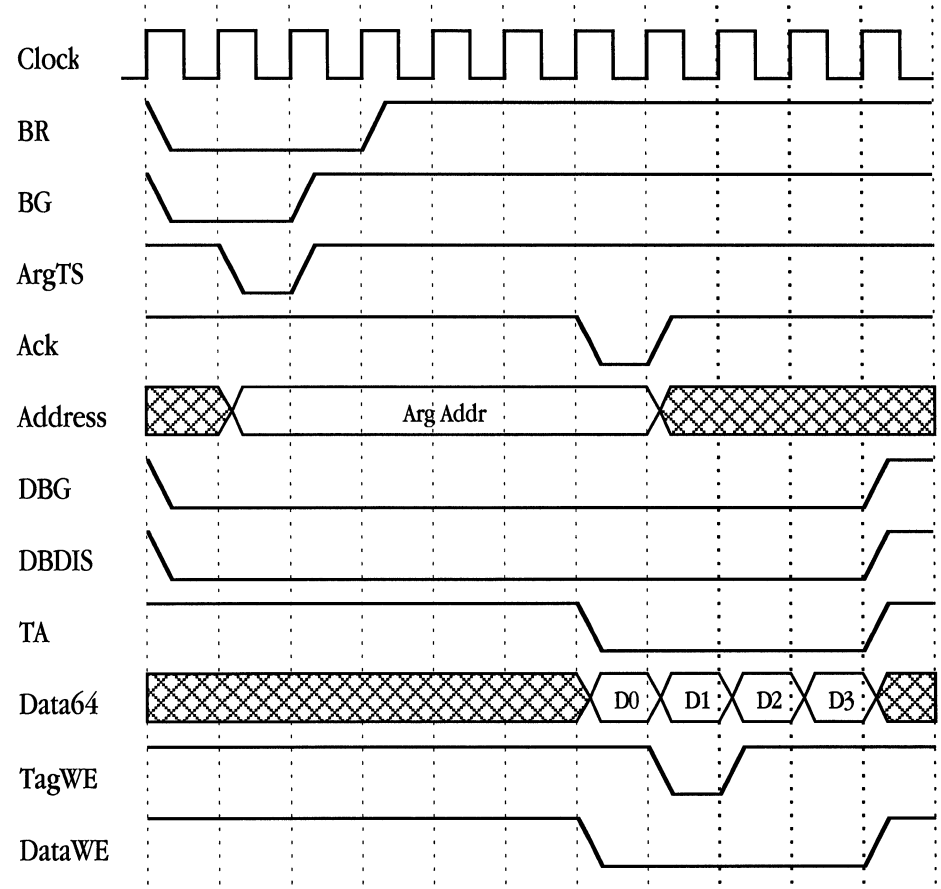


2.3.4.3 General flow of events for a µprocessor read miss with allocate (no dirty victim)

See Figure 9.

1. TS starts the lookup, DRAM cycle is started (RAS assertion etc.). Memory Controller asserts AACK.
2. TAG declares a miss, TAG allocate (with Dirty bit clear) occurs in ARTRY cycle. If the transaction receives an ARTRY, the TAG is not allocated (ARTRY gates TAGWE).
3. DRAMs are cycled, as each 8-byte chunk of data is transferred (TA asserted), it is written into the Cache SRAMs with the SRAM write enables asserted by the Cache Controller.

FIGURE 9 Read Miss With No Victim Timing Diagram

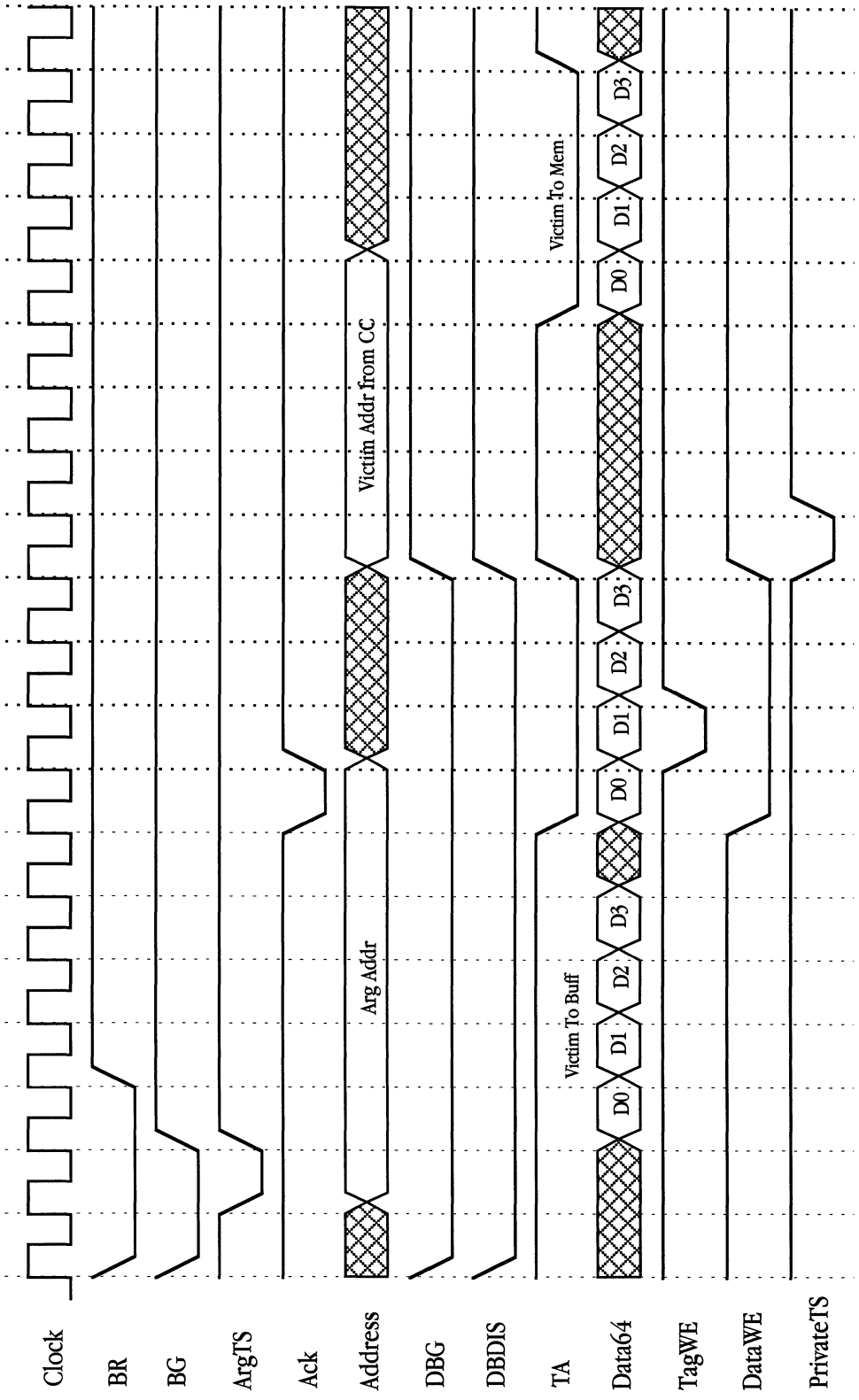


2.3.4.4 General flow of events for a µprocessor read miss allocate with a dirty victim

See Figure 10.

1. TS starts lookup, DRAM cycle is started (RAS assertion etc.). Memory Controller asserts AACK.
2. TAG declares a miss with dirty Victim. DRAM access is continued.
3. Cache Controller saves miss address (high-order bits) and reference (read or write).
4. Cache Controller reads out Dirty TAG from the TAG RAM and constructs the Victim address.
5. Cache Controller reads victim data out of SRAMs and saves in a victim buffer ('SaveVictim' signal asserted) along with address.
6. TAG is updated with new TAG, Valid set, Dirty clear.
7. DRAMs are cycled, as each 8-byte chunk of data is transferred (TA asserted), it is written into the Cache SRAMs with the SRAM write enables asserted by the Cache Controller.
8. Cache Controller asserts a 'private' TS and the Victim address to Memory Controller for a DRAM write. TS is not asserted to the µprocessor.
9. Cache Controller asserts 'GetVictim' and transfers the data from victim buffer to DRAM, asserting TA appropriately to Memory Controller.

FIGURE 10 Read Miss With Victim Timing Diagram

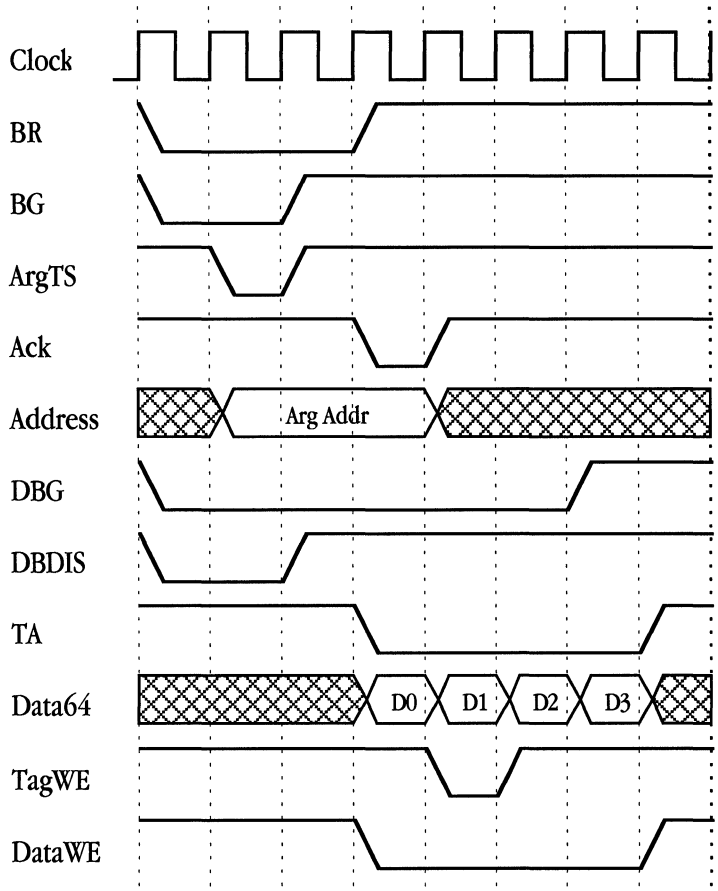


2.3.4.5 General flow of events for a µprocessor write hit

See Figure 11.

1. TS starts lookup, DRAM cycle is started (RAS assertion etc.). Memory Controller asserts AACK
2. TAG declares a hit. The Cache Controller asserts L2HIT to Memory Controller.
3. DRAM cycle is aborted gracefully.
4. Data is written into the Cache RAMs. The Cache Controller asserts TA and the SRAM write enables appropriately.

FIGURE 11 Write Hit Timing Diagram



2.3.4.6 General flow of events for a µprocessor write miss with allocate (no dirty victim)

See Figure 11 (same as write hit).

1. TS starts the lookup, DRAM cycle is started (RAS assertion etc.). Memory Controller asserts AACK.
2. TAG declares a miss, TAG allocate (with Dirty bit set) occurs in ARTRY cycle. If the transaction receives an ARTRY, the TAG is not allocated (ARTRY gates TAGWE).
3. Master data is written into the Cache RAMs. The Cache Controller asserts TA and the SRAM write enables appropriately.

2.3.4.7 General flow of events for a µprocessor write miss allocate with a dirty victim

See Figure 12.

1. TS starts lookup, DRAM cycle is started (RAS assertion etc.). Memory Controller asserts AACK
2. TAG declares a miss with dirty Victim. The Cache Controller asserts L2HIT to Memory Controller.
3. DRAM cycle is aborted gracefully.
4. Cache Controller reads out Dirty TAG from the TAG RAM and constructs the Victim address

5. Cache Controller reads victim data out of SRAMs and saves in a victim buffer along with address.
6. TAG is updated with new TAG, Valid set and Dirty set.
7. Master data is written into the Cache RAMs at the given index. The Cache Controller asserts a TA and the SRAM write enables appropriately
8. Cache Controller asserts TS and the Victim address to MC for a DRAM write.
9. Cache Controller asserts index to the cache SRAMs and transfers the data from SRAM to DRAM, asserting TA appropriately to MC only (TA block from asserting to the original Master).

FIGURE 12 Write Miss With Victim Timing Diagram

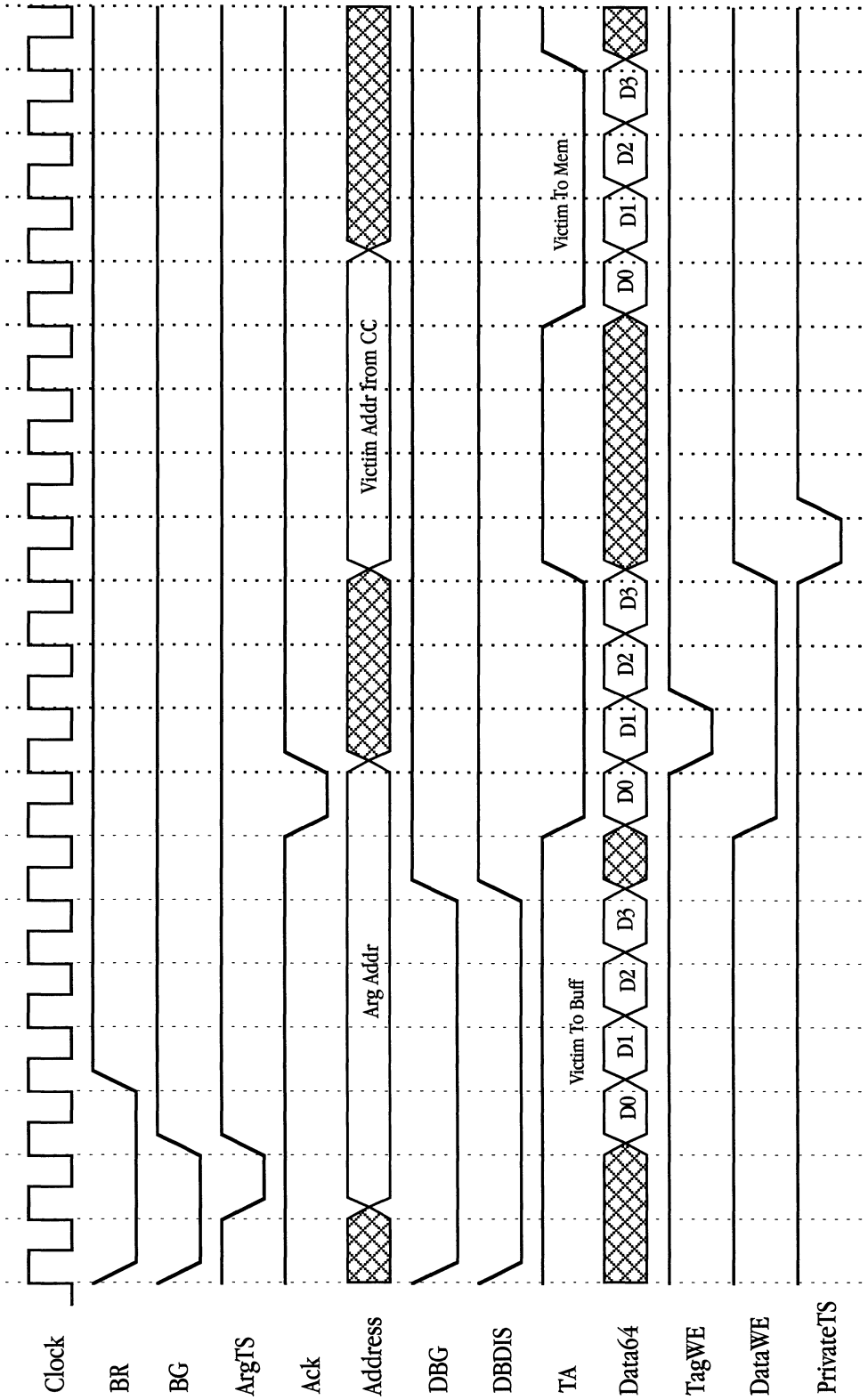


TABLE 10 Cache Behavior

ARBus Transaction	Cache Enable State	Cache Inhibit	TAG Lookup Result	Action	TAG After ARTRY
Any	Disabled	N/A	N/A	None	N/A
Any	Enabled	Asserted	N/A	None	Unchanged
D-Stream Read	I-Stream Only	N/A	N/A	None	Unchanged
I-Stream Read	D-Stream Only	N/A	N/A	None	Unchanged
D-Stream Write	I-Stream Only	N/A	N/A	None	Unchanged
D-Stream non-Burst Write or non-µP burst write	D-Stream	N/A	Invalid	DRAM written	Invalid
D-Stream non-Burst Write or non-µP burst write	D-Stream	N/A	Valid,Dirty No Match	DRAM written	Unchanged
D-Stream non-Burst Write or non-µP burst write	D-Stream	N/A	Valid,Dirty Match	Block Written to DRAM from SRAM, then original write completed to DRAM.	Invalid
D-Stream non-Burst Write or non-µP burst write	D-Stream	N/A	Valid,Dirty Match	Block invalidated, then original write completed to DRAM.	Invalid
I-Stream Read	I-Stream	False	Invalid	DRAM Access, Cache Filled with returned Data	Valid,Dirty Match
I-Stream Read	I-Stream	False	Valid,Dirty No Match	DRAM Access, Cache Filled with returned Data	Valid,Dirty Match
I-Stream Read	I-Stream	False	Valid,Dirty No Match	DRAM Access aborted, Victim written back to DRAM, DRAM Access, Cache Filled with returned Data	Valid,Dirty Match

TABLE 10 Cache Behavior

ARBus Transaction	Cache Enable State	Cache Inhibit	TAG Lookup Result	Action	TAG After ARTRY
I-Stream Read	I-Stream	False	Valid,Dirty Match	DRAM Access, aborted, data supplied from SRAM	Valid,Dirty Match
I-Stream Read	I-Stream	False	Valid,Dirty Match	DRAM Access, aborted, data supplied from SRAM	Valid,Dirty Match
D-Stream Read	D-Stream	False	Invalid	DRAM Access, Cache Filled with returned Data	Valid,Dirty Match
D-Stream Read	D-Stream	False	Valid,Dirty No Match	DRAM Access, Cache Filled with returned Data	Valid,Dirty Match
D-Stream Read	D-Stream	False	Valid,Dirty No Match	DRAM Access aborted, Victim written back to DRAM, DRAM Access, Cache Filled with returned Data	Valid,Dirty Match
D-Stream Read	D-Stream	False	Valid,Dirty Match	DRAM Access aborted, data supplied from SRAM	Valid,Dirty Match
D-Stream Read	D-Stream	False	Valid,Dirty Match	DRAM Access aborted, data supplied from SRAM	Valid,Dirty Match
D-Stream Write	D-Stream	False	Invalid	DRAM Access aborted, data written into SRAM	Valid,Dirty Match
D-Stream Write	D-Stream	False	Valid,Dirty No Match	DRAM Access aborted, data written into SRAM	Valid,Dirty Match
D-Stream Write	D-Stream	False	Valid,Dirty No Match	DRAM Access aborted, Victim written back to DRAM, data written into SRAM	Valid,Dirty Match
D-Stream Write	D-Stream	False	Valid,Dirty Match	DRAM Access aborted, data written into SRAM	Valid,Dirty Match
D-Stream Write	D-Stream	False	Valid,Dirty Match	DRAM Access aborted, data written into SRAM	Valid,Dirty Match

2.3.5 Non-CPU accesses

Table 10 lists the details of the behavior of the L2 cache in response to µprocessor-initiated ARBus transactions. The L2 cache is a resource that is also available to other system resources (Expansion and Video PCIs). In some cases, using the L2 cache for these accesses would be detrimental to system performance. The L2 Cache

Controller obeys the state of the CI wire so that other AGRBus masters can reference main memory without affecting the state of the L2 (and L1) cache. Some special case rules are required when ARBus masters other than the µprocessor reference the L2 cache:

1. When masters other than the µprocessor access the L2 cache, the transaction is eligible to receive an ARTRY assertion from the µprocessor. In these cases, the L2 cache cannot assert TA until the cycle following the ARTRY cycle.
2. Burst writes from non-µprocessor masters that hit in the L2 cache update the block. The block in the L2 data store is first written back to main memory (using the victim flow) and the non-µprocessor burst write is then simply forwarded on to main memory. The block in the L2 cache is left invalid. This behavior is similar to the non-burst write behavior.

2.4 ROM Access

ARBus addresses at reference ROM space are responded to by Hammerhead, cycling the ROMs using RomOE and Addr[27:28] to increment through burst reads. 8-Byte writes to ROM space are handled by using RomWE. All other length writes are not supported. It is the responsibility of software to meet any FLASH ROM timing requirements. The access time of the ROM can be programed by software for optimum performance (see Figure 20 on page 46).

2.5 Register Access

The registers described in Chapter 3 are accessed with 1-byte references to Hammerhead register space. Any other length access is not supported.

CHAPTER 3

Programmer's Model

The Hammerhead ASIC controls processor access to main memory (DRAM) addresses, ROM addresses and Hammerhead System Control Register accesses as defined by the TNT MacRISC address map shown in Table 11.

TABLE 11 TNT Memory Map

Start Address	End Address	Name	Comments
0x00000000	0x7FFFFFFF	Main Memory	2GB of main memory. Linearly accessed
0x80000000	0xBFFFFFFF	PCI Memory Space	1GB of PCI Memory Space. PCI cards are located here as well as all Frame Buffer space.
0xC0000000	0xEFFFFFFF	PCI Memory for NuBus SuperSlot Space	PCI Memory space used for NuBus Super Slots accessed via a PCI to NuBus Bridge. This space supports 3 Nubus Super Slots (C, D E). Addressing details are shown in the *TNT System ERS.*
0xF0000000	0xF1FFFFFF	PCI0 I/O Space	PCI0 in TNT contains the Video SubSystem. Addressing details are in the *TNT System ERS.*
0xF2000000	0xF3FFFFFF	PCI1 I/O Space	PCI1 in TNT contains the Expansion and I/O Sub-System. Addressing details are in the *TNT System ERS.*
0xF4000000	0xF5FFFFFF	PCI2 I/O Space	PCI2 is not used in TNT but is used in Tsunami as the second expansion PCI. Addressing details are in the *TNT System ERS.*
0xF6000000	0xF7FFFFFF	PCI3 I/O Space	PCI3 is not used in TNT.
0xF8000000	0xF8FFFFFF	System Control	System Control Registers are located here. CPU ID lives here.

TABLE 11 TNT Memory Map

Start Address	End Address	Name	Comments
0xF9000000	0xFEFFFFFF	PCI Memory for NuBus Slot Space	PCI Memory Space used for normal Nubus Slots. This space supports 6 Nubus Slots (9,A,B,C,D,E). Addressing details are in the *TNT System ERS.*
0xFF000000	0xFFFFFFFF	ROM Space	16 MB of Executable ROM space, See section 2.2.2

3.1 Memory-Like Space

Any size operand up to 32 bytes may be used to access Memory-Like space. Memory-like space is defined as DRAM, ROM and Built-In Frame Buffer space. Only DRAM and ROM space are kept coherent by the TNT hardware (Hammerhead and the 60x µprocessor). It is the responsibility of operating system software to maintain the cache coherence of the built-in Frame Buffer space and any expansion slot spaces.

3.1.1 DRAM

The Address Map allocates 2 GB of space for "DRAM" memory. This space is responded to on the ARBus by the Hammerhead ASIC. On all PCI/VCI bridge chips, memory transactions that appear on PCI0 or PCI1 with addresses that fall in this space are forwarded onto the ARBus for servicing by Hammerhead. DRAM memory in TNT is mapped starting at address zero. Up to 1.5GBytes of memory may be installed. DRAM memory is accessed contiguously.

3.1.2 ROM

The Address Map allocates 16MB of space for "ROM". This space is responded to on the ARBus by the Hammerhead ASIC. On all PCI/VCI bridge chips, memory transactions that appear on PCI0 or PCI1 with addresses that fall in this space are forwarded onto the ARBus for servicing by Hammerhead. ROM memory is mapped contiguously. At Reset, ROM is accessed starting at address 0xFFF00100.

3.2 System Control Registers

Hammerhead contains a number of system control registers to report the board and chip configurations. All of the registers are 8 bits wide and are accessed with 1-byte operands. These registers are mapped to exist on 16-byte boundaries as shown in Table 12.

TABLE 12 Hammerhead Register Addresses

ARBus Address	Register Name	Description	Reference
0xF8000000	CPUID	TNT CPU ID Field	Section 3.2.1 on page 40
0xF8000010	HHRev	Hammerhead Revision	Section 3.2.2 on page 41
0xF8000020	MBoardID	Motherboard ID	Section 3.2.3 on page 42
0xF8000030	CPUSpeed	CPU Speed	Section 3.2.4 on page 43
0xF8000040	MBDRAM	Motherboard DRAM configuration	Section 3.2.5 on page 44
0xF8000050	MemTime0	Memory Timing 0	Section 3.2.6 on page 45
0xF8000060	MemTime1	Memory Timing 1	Section 3.2.6 on page 45
0xF8000070	RefTime	Refresh Timing	Section 3.2.6 on page 45
0xF8000080	ROMTime	ROM Timing Register	Section 3.2.6 on page 45
0xF8000090	ArbConfig	Arbiter Configuration Register	Section 3.2.6 on page 45
0xF80000A0	TimeOut	ARBus Timeout Register	Section 3.2.10 on page 49
0xF80000B0	WhoAmI	Master-Dependent Return Value	Section 3.2.11 on page 50
0xF80000C0	IntReg	Secondary Processor Interrupt	Section 3.2.12 on page 51
0xF80000D0	TestMode	Functional Test Mode Control	Section 3.2.13 on page 52
0xF80000E0	L2Config	L2 Cache Configuration Register	Section 3.2.14 on page 53
0xF80000F0	L2 Diag	L2 Cache Diagnostic Register	Section 3.2.15 on page 55
0xF8000100	L2 TAG07	L2 Cache Tag bits [0:7] Register	Section 3.2.16 on page 56
0xF8000110	L2 TAG8F	L2 Cache Tag bits [8:F] Register	Section 3.2.16 on page 56
0xF8000120	L2 Debug	L2 Cache Debug Register	Section 3.2.17 on page 57
0xF80001C0	Bank0 Base MSB	MotherBoard Bank0 Base MSB	Section 3.2.18 on page 58
0xF80001D0	Bank0 Base LSB	MotherBoard Bank0 Base LSB	Section 3.2.18 on page 58
0xF80001E0	Bank1 Base MSB	MotherBoard Bank1 Base MSB	Section 3.2.18 on page 58
0xF80001F0	Bank1 Base LSB	MotherBoard Bank1 Base LSB	Section 3.2.18 on page 58
0xF8000200	Bank2 Base MSB	SIMM Bank2 Base MSB	Section 3.2.18 on page 58
0xF8000210	Bank2 Base LSB	SIMM Bank2 Base LSB	Section 3.2.18 on page 58
0xF8000220	Bank3 Base MSB	SIMM Bank3 Base MSB	Section 3.2.18 on page 58
0xF8000230	Bank3 Base LSB	SIMM Bank3 Base LSB	Section 3.2.18 on page 58
0xF8000240	Bank4 Base MSB	SIMM Bank4 Base MSB	Section 3.2.18 on page 58
0xF8000250	Bank4 Base LSB	SIMM Bank4 Base LSB	Section 3.2.18 on page 58
0xF8000260	Bank5 Base MSB	SIMM Bank5 Base MSB	Section 3.2.18 on page 58
0xF8000270	Bank5 Base LSB	SIMM Bank5 Base LSB	Section 3.2.18 on page 58

TABLE 12 Hammerhead Register Addresses

ARBus Address	Register Name	Description	Reference
0xF8000280	Bank6 Base MSB	SIMM Bank6 Base MSB	Section 3.2.18 on page 58
0xF8000290	Bank6 Base LSB	SIMM Bank6 Base LSB	Section 3.2.18 on page 58
0xF80002A0	Bank7 Base MSB	SIMM Bank7 Base MSB	Section 3.2.18 on page 58
0xF80002B0	Bank7 Base LSB	SIMM Bank7 Base LSB	Section 3.2.18 on page 58
0xF80002C0	Bank8 Base MSB	SIMM Bank8 Base MSB	Section 3.2.18 on page 58
0xF80002D0	Bank8 Base LSB	SIMM Bank8 Base LSB	Section 3.2.18 on page 58
0xF80002E0	Bank9 Base MSB	SIMM Bank9 Base MSB	Section 3.2.18 on page 58
0xF80002F0	Bank9 Base LSB	SIMM Bank9 Base LSB	Section 3.2.18 on page 58
0xF8000300	Bank10 Base MSB	SIMM Bank10 Base MSB	Section 3.2.18 on page 58
0xF8000310	Bank10 Base LSB	SIMM Bank10 Base LSB	Section 3.2.18 on page 58
0xF8000320	Bank11 Base MSB	SIMM Bank11 Base MSB	Section 3.2.18 on page 58
0xF8000330	Bank11 Base LSB	SIMM Bank11 Base LSB	Section 3.2.18 on page 58
0xF8000340	Bank12 Base MSB	SIMM Bank12 Base MSB	Section 3.2.18 on page 58
0xF8000350	Bank12 Base LSB	SIMM Bank12 Base LSB	Section 3.2.18 on page 58
0xF8000360	Bank13 Base MSB	SIMM Bank13 Base MSB	Section 3.2.18 on page 58
0xF8000370	Bank13 Base LSB	SIMM Bank13 Base LSB	Section 3.2.18 on page 58
0xF8000380	Bank14 Base MSB	SIMM Bank14 Base MSB	Section 3.2.18 on page 58
0xF8000390	Bank14 Base LSB	SIMM Bank14 Base LSB	Section 3.2.18 on page 58
0xF80003A0	Bank15 Base MSB	SIMM Bank15 Base MSB	Section 3.2.18 on page 58
0xF80003B0	Bank15 Base LSB	SIMM Bank15 Base LSB	Section 3.2.18 on page 58
0xF80003C0	Bank16 Base MSB	SIMM Bank16 Base MSB	Section 3.2.18 on page 58
0xF80003D0	Bank16 Base LSB	SIMM Bank16 Base LSB	Section 3.2.18 on page 58
0xF80003E0	Bank17 Base MSB	SIMM Bank17 Base MSB	Section 3.2.18 on page 58
0xF80003F0	Bank17 Base LSB	SIMM Bank17 Base LSB	Section 3.2.18 on page 58
0xF8000400	Bank18 Base MSB	SIMM Bank18 Base MSB	Section 3.2.18 on page 58
0xF8000410	Bank18 Base LSB	SIMM Bank18 Base LSB	Section 3.2.18 on page 58
0xF8000420	Bank19 Base MSB	SIMM Bank19 Base MSB	Section 3.2.18 on page 58
0xF8000430	Bank19 Base LSB	SIMM Bank19 Base LSB	Section 3.2.18 on page 58
0xF8000440	Bank20 Base MSB	SIMM Bank20 Base MSB	Section 3.2.18 on page 58
0xF8000450	Bank20 Base LSB	SIMM Bank20 Base LSB	Section 3.2.18 on page 58
0xF8000460	Bank21 Base MSB	SIMM Bank21 Base MSB	Section 3.2.18 on page 58

TABLE 12 Hammerhead Register Addresses

ARBus Address	Register Name	Description	Reference
0xF8000470	Bank21 Base LSB	SIMM Bank21 Base LSB	Section 3.2.18 on page 58
0xF8000480	Bank22 Base MSB	SIMM Bank22 Base MSB	Section 3.2.18 on page 58
0xF8000490	Bank22 Base LSB	SIMM Bank22 Base LSB	Section 3.2.18 on page 58
0xF80004A0	Bank23 Base MSB	SIMM Bank23 Base MSB	Section 3.2.18 on page 58
0xF80004B0	Bank23 Base LSB	SIMM Bank23 Base LSB	Section 3.2.18 on page 58
0xF80004C0	Bank24 Base MSB	SIMM Bank24 Base MSB	Section 3.2.18 on page 58
0xF80004D0	Bank24 Base LSB	SIMM Bank24 Base LSB	Section 3.2.18 on page 58
0xF80004E0	Bank25 Base MSB	SIMM Bank25 Base MSB	Section 3.2.18 on page 58
0xF80004F0	Bank25 Base LSB	SIMM Bank25 Base LSB	Section 3.2.18 on page 58

3.2.1 CPUID Register

This register contains a value that is specific to the TNT ASIC set.

FIGURE 13 TNT CPUID Register

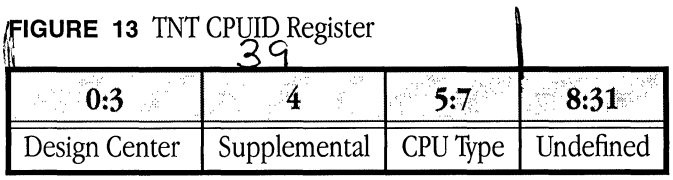


TABLE 13. TNT CPUID Register Description

Field	Bits	Type	Initial State	Description
Design Center	0:3	Read Only	0x3	This value defines this machine to be from the High-End RISC Design Center.
Supplemental Register	4	Read Only	0b1	This bit, if set, indicates that supplemental ID information appears in another register.
CPU Type	5:7	Read Only	0x1	This value defines this machine to be a TNT machine.
Undefined	8:31	Read Only	Undefined	

3.2.2 Hammerhead Revision Register

This register contains revision information that is specific to the given Hammerhead implementation.

FIGURE 14 Hammerhead Revision Register 61

0:3	4:7	8:31
RevNum	HHType	Undefined

TABLE 14 Hammerhead Revision Register Description

Field	Bits	Type	Initial State	Description
RevNum	0:3	Read Only	Silicon Specific	This field specifies the revision level for the Hammerhead ASIC
HHType	4:7	Read Only	Silicon Specific	This field specifies the feature set supported in this Hammerhead as follows: [4]: SDRAM Support: 0-> No, 1-> Yes [5]: L2 Cache: 0->Yes 1-> No [6]: Memory Width: 0-> 128 bits, 1-> 64 bits [7]: Slave Nodes: 0 -> 2 Bridges, 1->3 Bridges

Note: The Hammerhead 5.0 ASIC for TNT contains 0x51 in this register

F8000020

3.2.3 Hammerhead MotherBoard ID Register

9500-40

FIGURE 15 TNT Motherboard ID Register

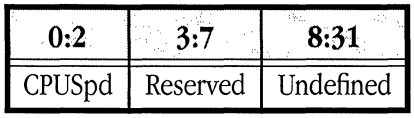
0:2	3:7	8:31
MBID	Reserved	Undefined

TABLE 15. TNT Motherboard ID Register Description

Field	Bits	Type	Initial State	Description
MBID	0:2	Read Only	From MuxAdr[0:2]	This field is taken from the state of the MuxAdr[0:2] pins at the de-asserting edge of reset to identify the type of Motherboard and resulting ASIC configuration. PCI1 is always assumed to exist. The bit assignment is, in general: MBID[0] -> VCI0 (PCI0) present MBID[1] -> PCI2 present MBID[2] -> Reserved Current defined legal values are: 0b000 -> TNT Proto-2 0b100 -> TNT EVT1 0b010 -> Tsunami EVT1
BurstRom	3	Read Only	From RomAdr[27]	This field is taken from the state of the RomAdr[27] pin at the de-asserting edge of reset to indicate that burstable roms have been installed in the system. Values are: 0b0: Burstable rom is not installed. 0b1: Burstable rom is installed.
Reserved	4:7	Read Only	0x00	
Undefined	8:31	Read Only	Undefined	

3.2.4 Hammerhead CPUSpeed Register

FIGURE 16 TNT CPUSpeed Register



0:2	3:7	8:31
CPUSpd	Reserved	Undefined

TABLE 16. TNT Motherboard ID Register Description

Field	Bits	Type	Initial State	Description
CPUSpd	0:2	Read Only	From Mux-Adr[3:5]	This field is taken from the state of the MuxAdr[3:5] pins at the de-asserting edge of reset to identify the speed of the µprocessor and the resultingARBus speed. Legal values are: 0b000 -> 40 MHz ARBus. 0b001 -> 33 MHz ARBus. 0b010 -> 44 MHz ARBus. 0b011 -> 50 MHz ARBus. others -> Reserved for the future.
Reserved	3:7	Read Only	0x00	
Undefined	8:31	Read Only	Undefined	

3.2.5 MotherBoard DRAM Configuration

This register contains information about the size of the DRAMs soldered to the motherboard.

FIGURE 17 MotherBoard DRAM Configuration Register

0:2	3:5	6:7	8:31
MBSize	Reserved	MemConfig	Undefined

TABLE 17. MotherBoard DRAM Configuration Register Description

Field	Bits	Type	Initial State	Description
MBSize	0:2	Read Only	From Mux-Adr[6:8]	This field is taken from the state of the MuxAdr[8:10] pins at the de-asserting edge of reset to identify the size of the motherboard DRAMs. Legal values are: 0b000 -> Sixteen 512Kx8 DRAMs 0b001 ->Thirty-two 1Mx4 DRAMs Others -> Reserved
Reserved	3:5	Read Only	0x00	
MemConfig	6:7	Read Only	From Mux-Adr[9:10]	This field is taken from the state of the MuxAdr[6:7] pins at the de-asserting edge of reset to identify the configuration of DRAMs. Legal values are: 0b00 -> 128-Bit DRAM. 0b01 -> 64-Bit DRAM. else ->Reserved.
Undefined	8:31	Read Only	Undefined	

3.2.6 Memory Timing Registers

These registers allow software to program the optimum DRAM timing for a given ARBus Speed. For typical settings, please see Table 20 on page 46.

F800 0250

FIGURE 18 Memory Timing Register 0

0:2	3:5	6	7	8:31
RasTime	RasPCTime	DelayRdTA	Reserved	Undefined

TABLE 18. Memory Timing Register 0 Description

Field	Bits	Type	Initial State	Description
RasTime	0:2	R/W	0b100	This field defines the number of ARBus clock cycles for minimum active RAS.
RasPCTime	3:5	R/W	0b011	This field defines the minimum number of ARBus clock cycles for RAS pre-charge time.
DelayRdTA	6	R/W	0b1	This bit, if set, forces an additional ARBus clock cycle delay in asserting the first TA of a DRAM access.
Reserved	7	Read Only	0b0	
Undefined	8:31	Read Only	Undefined	

FIGURE 19 Memory Timing Register 1

F800 0060

0:2	3:5	6:7	8:31
WrCasTime	RdCasTime	Reserved	Undefined

TABLE 19. Memory Timing Register1 Description

Field	Bits	Type	Initial State	Description
WrCasTime	0:2	R/W	0b010	This field defines the number of ARBus clock cycles for active CAS time for writes.
RdCasTime	3:5	R/W	0b011	This field defines the number of ARBus clock cycles for active CAS time for reads.
Reserved	6:7	Read Only	0b00	
Undefined	8:31	Read Only	Undefined	

TABLE 20 Suggested Memory Timing Register Values

ARBus Speed	60ns DRAM	70ns DRAM	80ns DRAM
33 MHz	MemTime0->0x48 MemTime1->0x28	MemTime0->0x68 MemTime1->0x28	MemTime0->0x68 MemTime1->0x28
40 MHz	MemTime0->0x6A MemTime1->0x28	MemTime0->0x6A MemTime1->0x28	MemTime0->0x8E MemTime1->0x48
44 MHz	MemTime0->0x6A MemTime1->0x28	MemTime0->0x8E MemTime1->0x48	MemTime0->0x8E MemTime1->0x4C
50 MHz	MemTime0->0x6A MemTime1->0x28	MemTime0->0x8E MemTime1->0x4C	MemTime0->0x8E MemTime1->0x4C

3.2.7 Refresh Timing Register

This register allows software to program the optimum DRAM Refresh timing for the given ARBus speed.

FIGURE 20 Refresh Timing Register

F800 0070

0:7	8:31
RefTime	Undefined

TABLE 21. Refresh Timing Register Description

Field	Bits	Type	Initial State	Description
RefTime	0:7	R/W	0x61	This field defines the most significant 8 bits of a 10-bit refresh counter. When this counter expires, a refresh of DRAM is scheduled. Refresh is required every 15.6µseconds. For optimum performance, this register should be programmed as follows: ARBus Speed RefTime Value 33 MHz-------> 0x82 40 MHz-------> 0x9C 44 MHz-------> 0xAB 50 MHz-------> 0xC3
Undefined	8:31	Read Only	Undefined	

3.2.8 ROM Timing Register

This register allows software to program the optimum ROM timing for the given ARBus speed.

FIGURE 21 ROM Timing Register

0:3	4:6	7	8:31
FirstWd	BurstRate	RomWE	Undefined

TABLE 22. ROM Timing Register Description

Field	Bits	Type	Initial State	Description
FirstWd	0:3	R/W	0b1000	This field defines the minimum number of ARBus clock cycles between the assertion of TS and the first assertion of TA for ROM reads and writes. For optimum performance after checking MotherboardID.BurstROM (see Section 3.2.3 on page 42), this field should be programmed as follows: 33 MHz --->0x4 non-burstable, 0x5 burstable. 40 MHz --->0x5 non-burstable, 0x6 burstable. 44 MHz --->0x6 non-burstable, 0x7 burstable. 50 MHz --->0x7 non-burstable, 0x8 burstable.
BurstRate	4:6	R/W	0b100	This field defines the minimum number of ARBus clock cycles between TA assertions. For optimum performance after checking MotherboardID.BurstROM (see Section 3.2.3 on page 42), this field should be programmed as follows: 33 MHz ---> 0x5 non-burstable, 0x4 burstable. 40 MHz ---> 0x6 non-burstable, 0x4 burstable. 44 MHz ---> 0x6 non-burstable, 0x5 burstable. 50 MHz ---> 0x7 non-burstable, 0x5 burstable.
RomWE	7	R/W	0b0	This bit enables ROM flashing. Any write to ROM space will not result in a ROM flash without first setting this bit.
Undefined	8:31	Read Only	Undefined	

3.2.9 Arbiter Configuration Register

This register allows software to enable and select the Hammerhead ARBus Arbiter to park a given master, removing 2 ARBus cycles from that masters memory latentcy in most cases.

FIGURE 22 Arbiter Configuration Register

0	1:3	4	5	6	7	8:31
ParkEna	ParkSel	ModPPC	PipeEna	TwoCPU	Reserved	Undefined

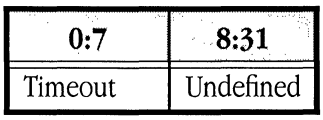
TABLE 23. Arbiter Configuration Register Description

Field	Bits	Type	Initial State	Description
ParkEna	0	R/W	0b0	When set, this bit forces the arbiter to park (assert BG) to the master selected in the ParkSel field.
ParkSel	1:3	R/W	0b0	This field defines which ARBus master has its BG asserted during idle ARBus cycles if ParkEna is set. ParkSel is encoded as follows: 0b000 -> Kaos/Control (Video Bridge) Parked 0b001 -> Bandit 1 (GC/Expansion Bandit) Parked 0b010 -> Bandit 2 (Extra Expansion Bandit) Parked 0b011 -> Primary Processor Parked 0b100 -> Secondary Processor Parked
ModPPC	4	R/W	0b0	Modern PowerPC. When Set, this bit indicates that the µprocessor is a modern PowerPC chip and includes the "Fast L2 Cache" modes. When this bit is clear, Hammerhead assumes the µprocessor is a 601 and uses DBG appropriately. When this bit is set, DBG to the primary µprocessor is asserted immediately and DBDis is used to control the processor write data assertions.
Pipe Enable	5	R/W	0b0	This bit no longer controls anything, but is still read/writable for backward software compatibility.
TwoCPU	6	Read Only	0b0	If the Hammerhead arbiter ever sees a Bus Request from the secondary CPU, this bit is set. Therefore, in mono-processor systems, this bit will always be clear. In Stereo systems, this bit will be set.
Reserved	7	Read Only	0b0	
Undefined	8:31	Read Only	Undefined	

3.2.10 ARBus Timeout Register

This register allows software to program the ARBus timeout value. The Hammerhead ARBus Arbiter starts a counter whenever TS is asserted and stops it whenever an AACK is asserted. If the counter expires before the AACK occurs, the ARBus Arbiter asserts AACK and DBG, and immediately asserts TA and TEA to signal a transfer error to the Master. If the ARBus Master was the 60x µprocessor, a machine check is generated.

FIGURE 23 ARBus Timeout Register



0:7	8:31
Timeout	Undefined

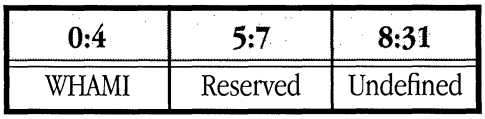
TABLE 24 ARBus Timeout Register Description

Field	Bits	Type	Initial State	Description
Timeout	0:7	R/W	0x00	This field defines the most significant bits of a 14-bit timeout counter. A larger value in this field yields a longer timeout. A value of 0x00 is the largest value (256 decimal). The minimum and maximum timeout values are: 33 MHz ---> Minimum 1.92 µs, Maximum 491 µs. 40 MHz ---> Minimum 1.60 µs, Maximum 409 µs. 44 MHz ---> Minimum 1.45 µs, Maximum 372 µs. 50 MHz ---> Minimum 1.28 µs, Maximum 327 µs.
Undefined	8:31	Read Only	Undefined	

3.2.11 Who Am I Register

This register allows software to determine which CPU in a dual processor system the software is running on. This register simply returns the value of the current master.

FIGURE 24 WHAMI Register



0:4	5:7	8:31
WHAMI	Reserved	Undefined

TABLE 25 WHAMI Register Description

Field	Bits	Type	Initial State	Description
WHAMI	0:4	Read Only	0x00	This field returns the value of the current bus master: 0b10000 --> Kaos/Control (Video Bridge) 0b01000 --> PCI Bridge 1 0b00100 --> PCI Bridge 2 0b00010 --> Primary CPU 0b00001 --> Secondary CPU
Reserved	5:7	Read Only	0x00	
Undefined	8:31	Read Only	Undefined	

3.2.12 Secondary Interrupt Register

This register allows software to assert the interrupt pin on the secondary µprocessor.

FIGURE 25 Secondary Interrupt Register

0	1:7	8:31
SecInt	Reserved	Undefined

TABLE 26 Secondary Interrupt Register Description

Field	Bits	Type	Initial State	Description
SecInt	0	R/W	0b1	This bit drives the secondary µprocessor interrupt pin directly. On the 601, 603 and 604 µprocessors, this pin is active low, hence this bit is reset to 1. Setting this bit to 0 asserts the interrupt. Setting this bit to 1 clears the interrupt.
Reserved	1:7	Read Only	0x00	
Undefined	8:31	Read Only	Undefined	

F800 0000

3.2.13 Test Mode Register

This register allows diagnostic software to set some of the large counters in Hammerhead to overflow (expire) in shorter time intervals for testing purposes.

FIGURE 26 Test Mode Register

0	1:7	8:31
RefrTest	Reserved	Undefined

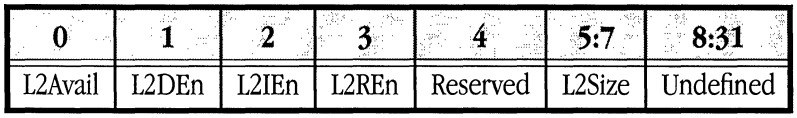
TABLE 27 Test Mode Register Description

Field	Bits	Type	Initial State	Description
RefrTest	0	R/W	0b0	When set, this bit forces the DRAM refresh counter (Section 3.2.6 on page 45) to overflow every 64 cycles, thereby queueing a refresh.
Reserved	1:7	Read Only	0x00	
Undefined	8:31	Read Only	Undefined	

3.2.14 L2 Cache Configuration Register

This register controls the behavior of the Hammerhead L2 Cache Controller.

FIGURE 27 L2 Cache Configuration Register



0	1	2	3	4	5:7	8:31
L2Avail	L2DEn	L2IEn	L2REn	Reserved	L2Size	Undefined

TABLE 28. L2 Cache Configuration Register Description

Field	Bits	Type	Initial State	Description
L2 Cache Available	0	Read Only	From Mux-Adr[11]	If set, this bit indicates that a second level cache is present. This bit reflects the state of the MuxAdr[11] pin on the de-asserting edge of Reset.
L2 D-Stream Enable	1	R/W	0b0	Second Level Cache D-Stream Access Enable: If set, this bit forces the Memory Controller to respond to D-Stream memory references with cache hits and allocate in the L2 Cache on D-Stream misses. If both L2IEn and L2DEn are clear the L2 Cache is disabled.
L2 I-Stream Enable	2	R/W	0b0	Second Level Cache I-Stream Access Enable: If set, this bit forces the Memory Controller to respond to I-Stream memory references with cache hits and allocate in the L2 Cache on I-Stream misses. If both L2IEn and L2DEn are clear the L2 Cache is disabled.
L2 ROM Enable	3	R/W	0b0	Second Level Cache ROM Access Enable: If set, this bit forces the Memory Controller to respond to ROM address space with cache hits and allocate in the L2 cache on misses to ROM address space. This bit is valid only if L2IEn or L2DEn is set.
Reserved	4	Read Only	0b0	
L2 Size	5:7	Read Only	From WrtClkEn[0:1]	This field specifies the size of the L2 Cache installed. This field is decoded as follows: 0b000 ->512K 0b001 ->256K 0b010 ->1Meg 0b011 ->4Meg others ->Reserved

To enable the Hammerhead L2 cache, software must perform the following sequence of operations:

1. Set L2 Cache Diag [0] and clear L2 Cache Diag [1].
2. Set any of L2 Cache Configuration Register [1:3] as desired
3. Starting at any DRAM address, read a L2 Size number of bytes.
4. Clear L2 Cache Diag[0].

3.2.15 L2 Cache Diagnostic Register

This register allows software to set and clear the Valid and Status bits in the Tag Store for cache invalidate and diagnostic functions.

FIGURE 28 L2 Cache Diagnostic Register

0	1	2	3:7	8:31
DiagEna	DiagValid	DiagDirty	Reserved	Undefined

TABLE 29. L2 Cache Diagnostic Register Description

Field	Bits	Type	Initial State	Description
Diagnostic Enable	0	R/W	0b0	If set, this bit forces the L2 cache into a diagnostic mode, preventing the cache from responding to any ARBus Request, irrespective of L2Config.L2Ena. When this bit is set, any memory space access loads the values found in DiagValid and DiagDirty into the Tag Store RAM at the index given by the ARBus Address.
Diagnostic Valid	1	R/W	0b0	When L2Diag.DiagEna is set, the value in this field is loaded into the Tag Store Valid bit at the index given in the ARBus Address.
Diagnostic Dirty	2	R/W	0b0	When L2Diag.DiagEna is set, the value in this field is loaded into the Tag Store Dirty bit at the index given in the ARBus Address.
Reserved	3:7	Read Only	0	

3.2.16 L2 Cache TAG Registers

This read-only register allows software to observe the contents of the Tag Store RAM. following each memory or ROM space access. The contents of the TAG ram at the given index is latched into these registers. The TNT Tag store is always 16 bits wide with the Valid bit existing in bit 15 and the Dirty bit existing in bit 14. The significance of the remainder of the bits is a function of the cache size and resulting Tag width as shown in Table 7 on page 21.

FIGURE 29 L2 Cache TAG07 Register

F8 00 0100

0:7	8:31
TAG[0:7]	Undefined

TABLE 30. L2 Cache TAG815Register

F8 00 0110

0:7	8:31
TAG[8:15]	Undefined

F800 0120

3.2.17 L2 Debug Register

This register allows software to observe the state of critical Hammerhead internal states to aid in debug.

FIGURE 30 L2 Cache Debug Register

0	1	2	3	4	5:7	8:31
L2Hit	L2 Victim	L2Write	L2Allocate	L2Burst	Reserved	Undefined

TABLE 31. L2 Cache Debug Register Description

Field	Bits	Type	Initial State	Description
L2Hit	0	Read Only	Undefined	If set, this bit indicates that the most recent Tag lookup yielded a hit condition.
L2 Victim	1	Read Only	Undefined	If set, this bit indicates that the most recent Tag lookup required a victim to be written back to DRAM.
L2 Write	2	Read Only	Undefined	If set, this bit indicates that the most recent ARBus transaction that accessed the cache was a write.
L2 Allocate	3	Read Only	Undefined	If set, this bit indicates that the most recent ARBus transaction that accessed the cache required an allocate.
L2 Burst	4	Read Only	Undefined	If set, this bit indicates that the most recent ARBus transaction that accessed the cache was a burst.
Reserved	5:7	Read Only	0	

3.2.18 DRAM Base Registers

The Hammerhead DRAM controller allows for very flexible configurations of on-board DRAM and DRAM SIMMs. To properly configure a system, each bank of DRAMs is assigned a base address with a pair of registers that, concatenated together, form an eleven bit register that defines the base address, the addressing mode and if a given bank is interleaved with its companion.

Twenty-Six Pairs of these registers are implemented to control 2 banks of on-board DRAM and 24 banks of expansion DRAM SIMMs (12 double-sided SIMM sockets). Figure 3 on page 11 illustrates how the banks are organized.

FIGURE 31 Bankn Base Register LSB

0:7	8:31
Base Addr [2:9]	Undefined

FIGURE 32 Bankn Base Register MSB

0:4	5	6	7	8:31
Reserved	Interleave Enable	Address Mode	Base Addr [1]	Undefined

TABLE 32. Bankn Base Register Description

Field	Bits	Type	Initial State	Description
Interleave Enable	MSB[5]	R/W	Undefined	This bit enables a given set of banks to be considered an interleave set, 128-bits wide. This bit is only implemented for the Even banks. For Bank0, this bit is always set (i.e. Bank0 and Bank1, the motherboard banks, are always interleaved).
Address Mode	MSB[6]	R/W	Undefined	This bit configures the mapping of ARBus address bits to DRAM RAS/CAS address bits to enable support of all configurations of 1, 4, 16, and 64MBit DRAMs (seeTable 33 on page 59).
Base Addr	{MSB[7], LSB[0:7]}	R/W	Undefined	This field is used as the base address that ARBus addresses are compared to. If ARBus address bits [1:9] are greater than or equal to this field AND less then the value of BankN+1 Base, the given bank is selected.

TABLE 33 Address ModeBit Settings for Various DRAMs

DRAM Type	Row Bits	Column Bits	Mode Bit setting
1MBit 1Mx1	10	10	1
1MBit 256Kx4	9	9	1
4M Bit 4Mx1	11	11	1
4M Bit 1Mx4	10	10	1
4M Bit 512Kx8	10	9	1
4M Bit 256Kx16	10	8	0
16M Bit 16Mx1	12	12	0
16M Bit 4Mx4	11	11	1
16M Bit 4Mx4	12	10	1
16M Bit 2Mx8	11	10	1
16M Bit 2Mx8	12	9	0
16M Bit 1Mx16	12	8	0
64M Bit 16Mx4	12	12	0
64M Bit 8Mx8	12	11	0
64M Bit 4Mx16	11	11	1
64M Bit 4Mx16	12	10	1
64M Bit (other)	>12	----	Not Supported

TABLE 34 ARBus Address to DRAM Address Mapping - Non-interleaved Banks

Address Mode	DR0	DR1	DR2	DR3	DR4	DR5	DR6	DR7	DR8	DR9	DR10	DR11
AddrMode=0 RAS Address	AR9	AR10	AR11	AR12	AR13	AR14	AR15	AR16	AR17	AR18	AR19	AR20
AddrMode=0 CAS Address	AR5	AR6	AR7	AR8	AR21	AR22	AR23	AR24	AR25	AR26	AR27	AR28
AddrMode=1 RAS Address	AR7	AR8	AR10	AR11	AR12	AR13	AR14	AR15	AR16	AR17	AR18	AR19
AddrMode=1 CAS Address	AR6	AR7	AR9	AR20	AR21	AR22	AR23	AR24	AR25	AR26	AR27	AR28

TABLE 35. ARBus Address to DRAM Address Mapping - Interleaved Banks

Address Mode	DR0	DR1	DR2	DR3	DR4	DR5	DR6	DR7	DR8	DR9	DR10	DR11
AddrMode=0 RAS Address	AR8	AR9	AR10	AR11	AR12	AR13	AR14	AR15	AR16	AR17	AR18	AR19
AddrMode=0 CAS Address	AR4	AR5	AR6	AR7	AR20	AR21	AR22	AR23	AR24	AR25	AR26	AR27
AddrMode=1 RAS Address	AR6	AR7	AR9	AR10	AR11	AR12	AR13	AR14	AR15	AR16	AR17	AR18
AddrMode=1 CAS Address	AR5	AR6	AR8	AR19	AR20	AR21	AR22	AR23	AR24	AR25	AR26	AR27

3.2.18.1 Memory Sizing and Stitching Algorithm

To properly configure the Hammerhead DRAM controller, the following algorithms should be used. Banks are sized by checking the last location in each 1MByte chunk to determine the bank size.

FIGURE 33 Bank Stitching Algorithm

Assume BanknBase[0:15] <= {5'b0,BanknMSB,BanknLSB}

1. Set each BanknBase to be 16'b0000 000n nnnn 0000, where n is the bank number (in binary), and size the bank (Address mode bit clear). If the bank size is not a power of 2 (i.e. a size of 5MB or 13MB), goto step 2. If the bank size is a power of 2, goto step 3.
2. Set each BanknBase to be 16'b0000 0001n nnnn 0000, where n is the bank number (in binary), and size the bank (Address mode bit set).
3. Program the base address field for each bank with bits [1:9] of the first address in the bank. The first address of a given bank is a function of the sum of the sizes of all lower numbered banks.
4. If a given set of Banks (01,23 etc.) are the same size AND their addressing modes are identical, additionally set the interleave bit in the even bank base register.

CHAPTER 4

Interface Specification

This chapter describes the Hammerhead I/O pins and their relation to the rest of the system.

4.1 Hammerhead Pins

Table 36 Lists the I/O Pins on the Hammerhead ASIC.

TABLE 36 Hammerhead I/O Pins

Mnemonic	Direction	Assertion	Description
ARBus Signals			
AD[0:31]	In	High	ARBus Address
AACK	In	Low	Address Acknowledge
ARTRY	In	Low	Address Retry from the 60x µprocessor
BR[0:4]	In	Low	Bus Request from Masters 0 ->Kaos/Control (Video Bridge) 1 ->Bandit 1 2 ->Bandit 2 3->Primary µProcessor 4-> Secondary µProcessor
BG[0:4]	Out	Low	Bus Grant to Masters. Same mapping as BR.
DBG[0:4]	Out	Low	Data Bus Grant to Masters. Same mapping as BR.
DBWO	Out	Low	Data Bus Write Only, wired to the µProcessors. This signal, when asserted with DBG, allows writes to be reordered around reads if needed. This is used for deadlock avoidance in the presence of snoop pushes.

TABLE 36 Hammerhead I/O Pins

Mnemonic	Direction	Assertion	Description
RdDA[0:2]	In	Low	Read Data Available from the External Slaves. 0 -> Kaos/Control (Video Bridge) 1 -> Bandit 1 2 -> Bandit 2 Also RdDA[2] is used for scan data in.
SSD[0:2]	Out	Low	Slave Source/Sink Data to the External Slaves. Same mapping as RdDA. Also, SSD[2] is used for scan data out.
SACK[0:2]	In	Low	Slave Acknowledge from the external Slaves. Same mapping as RdDA.
DBDIS	Out	Low	DataBus Disable to the 603 and 604 primary processor.
DT[0:7]	I/O	High	Most Significant Byte of ARBus data for register access.
DataDir	Out	High	Used to control an external transceiver for internal register accesses. This is so that the transceiver can be placed directly in line with the data bus on a motherboard. Note that its use is not mandatory. Simply hooking Hammerhead up to ARBus data will work fine functionally, but this may require prohibitively long data bus board traces. 0 -> ARBus Data points toward Hammerhead. 1 -> Hammerhead Data points toward ARBus.
TA	I/O	Low	Transfer Acknowledge
TEA	I/O	Low	Transfer Error Acknowledge
TBST	In	Low	Transfer Burst
TSIZ[0:2]	In	High	Transfer Size
TS	In	Low	Transfer Start
TT[1],TT[3]	In	High	Transaction Type
TC[0]	In	High	Transfer Code. If asserted, ARBus read is an I-Stream fetch, else D-Stream.
CI	In	Low	If asserted, this transaction is cache-inhibited.
Memory Interface Pins			
Ras[1:12]	Out	Low	DRAM row address strobe.
Cas[0:15]	Out	Low	DRAM column address strobe.
Wrt	Out	Low	DRAM write enable.
MuxAdr[0:11]	I/O	High	DRAM multiplexed address. Used as inputs during reset.

TABLE 36 Hammerhead I/O Pins

Mnemonic	Direction	Assertion	Description
RdOEn	Out	Low	Zax ARBus Output Enable.
RdHold	Out	Low	Zax Latch enable for grabbing DRAM data off of the DRAM Data Bus.
BankSel	Out	High	Zax Bank Select. When asserted, selects DRAM Data [64:127] to be asserted on the ARBus.
WrtOEn	Out	Low	Zax DRAM Data Output Enable.
WrtClkEn[0:1]	I/O	Low	Zax Clock enables for grabbing write data off of the ARBus. Used as inputs during reset.
WrtMuxSel[0:1]	Out	High	Zax write buffer selects for non-interleaved DRAM writes.
Cache Interface Pins			
TagALE	Out	Low	Tag address latch enable. When asserted, the address to the tag store will be frozen in an external latch.
Tag[0:13]	I/O	High	Tag Store Value.
TagValid	I/O	High	Valid bit from Tag Store.
TagDirty	I/O	High	Dirty Bit from Tag Store.
TagOEn	Out	Low	Tag Store Output Enable.
TagWEn	Out	Low	Tag Store Write Enable.
CacDOEn	Out	Low	Data Store Output Enable.
CacDWEn	Out	Low	Data Store Write Enable.
CacBurstAdv	Out	Low	Burst Advance to Data Store SRAMs.
CacXfrStart	Out	Low	Transfer Start to Data Store SRAMs.
ROM Interface Pins			
RomALE	Out	Low	Rom address latch enable. When asserted, the address to the ROM will be frozen in an external latch.
RomAddr[27:28]	Out	High	The two LSBs of address to the ROM. Sourced from Hammerhead so that bursts may occur.
RomDataOEn	Out	Low	ROM Output Enable
RomDataWEn	Out	Low	ROM Write Enable
Utility Pins			

TABLE 36 Hammerhead I/O Pins

Mnemonic	Direction	Assertion	Description
Clk	In	High	This signal is a copy of the system clock. The duty cycle is guaranteed to be 50%. The frequency is fixed and within the range of 25-50MHz. All synchronous transfers involving Hammerhead are referenced to this Clock.
Reset	In	Low	Active low signal indicating power-on reset has occurred.
TestMode	In	Low	Tri-states all pins for in-circuit testing.
Int	Out	Low	Secondary processor interrupt, controlled via a Hammerhead register.
ScanClk	In	High	Scan test clock.
ScanTest	In	Low	For scan testing only. Makes all internal latches transparent and turns off all active pad pullups for IDDQ testing.

4.2 Hammerhead Physical Pinout

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
1	VSS	GROUND				
2	TAGDIRTYPAD_H	Bidi	See Note 1	0.00	15.49	8/8 ma
3	TAGVALIDPAD_H	Bidi	See Note 1	0.00	15.13	8/8 ma
4	TAG13PAD_H	Bidi	See Note 1	0.00	13.87	8/8 ma
5	VSS	GROUND				
6	TAG12PAD_H	Bidi	See Note 1	0.00	13.85	8/8 ma
7	TAG11PAD_H	Bidi	See Note 1	0.00	13.94	8/8 ma

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
8	TAG10PAD_H	Bidi	See Note 1	0.00	13.93	8/8 ma
9	VDD	+5V POWER				
10	TAG09PAD_H	Bidi	See Note 1	0.00	13.93	8/8 ma
11	TAG08PAD_H	Bidi	See Note 1	0.00	13.87	8/8 ma
12	TAG07PAD_H	Bidi	See Note 1	0.00	13.84	8/8 ma
13	VSS	GROUND				
14	TAG06PAD_H	Bidi	See Note 1	0.00	13.82	8/8 ma
15	TAG05PAD_H	Bidi	See Note 1	0.00	13.82	8/8 ma
16	TAG04PAD_H	Bidi	See Note 1	0.00	13.78	8/8 ma
17	TAG03PAD_H	Bidi	See Note 1	0.00	13.79	8/8 ma
18	VSS	GROUND				
19	TAG02PAD_H	Bidi	See Note 1	0.00	13.78	8/8 ma
20	TAG01PAD_H	Bidi	See Note 1	0.00	13.77	8/8 ma
21	TAG00PAD_H	Bidi	See Note 1	0.00	13.75	8/8 ma
22	VSS	GROUND				
23	TAGOENPAD_L	Output	--------	--------	12.31	8/8 ma
24	TAGWENPAD_L	Output	--------	--------	13.27	8/8 ma
25	CACDOENPAD_L	Output	--------	--------	See Note 2	12/12 ma
26	CACDWENPAD_L	Output	--------	--------	15.10	12/12 ma
27	VDD	+5V POWER				
28	CACBURSTADVPAD_L	Output	--------	--------	11.31	12/12 ma
29	CACXFRSTARTPAD_L	Output	--------	--------	11.28	12/12 ma
30	DBDISPAD_L	Output	--------	--------	10.91	8/8 ma
31	VSS	GROUND				
32	CIPAD_L	Input	0.00	1.02	--------	--------
33	VDD	+5V POWER				
34	TC0PAD_H	Input	0.13	1.44	--------	--------
35	DATADIRPAD_H	Output	--------	--------	12.51	12/24 ma

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
36	VSS	GROUND				
37	RAS01PAD_L	Output	--------	--------	11.21	12/24 ma
38	RAS02PAD_L	Output	--------	--------	11.28	12/24 ma
39	RAS03PAD_L	Output	--------	--------	11.16	12/24 ma
40	VSS	GROUND				
41	RAS04PAD_L	Output	--------	--------	11.22	12/24 ma
42	RAS05PAD_L	Output	--------	--------	11.34	12/24 ma
43	RAS06PAD_L	Output	--------	--------	11.38	12/24 ma
44	VDD	+5V POWER				
45	RAS07PAD_L	Output	--------	--------	11.35	12/24 ma
46	RAS08PAD_L	Output	--------	--------	11.57	12/24 ma
47	RAS09PAD_L	Output	--------	--------	11.42	12/24 ma
48	VSS	GROUND				
49	RAS10PAD_L	Output	--------	--------	11.59	12/24 ma
50	RAS11PAD_L	Output	--------	--------	11.44	12/24 ma
51	RAS12PAD_L	Output	--------	--------	11.41	12/24 ma
52	VSS	GROUND				
53	VSS	GROUND				
54	WRTPAD_L	Output	--------	--------	11.10	12/24 ma
55	BANKSELPAD_H	Output	--------	--------	12.62	8/8 ma
56	RDOENPAD_L	Output	--------	--------	13.58	8/8 ma
57	WRTOENPAD_L	Output	--------	--------	13.41	8/8 ma
58	RDHOLDPAD_L	Output	--------	--------	12.35	8/8 ms
59	VSS	GROUND				
60	WRTCLKEN0PAD_L	Bidi	0.00	0.00	12.75	8/8 ma
61	WRTCLKEN1PAD_L	Bidi	0.00	0.00	12.78	8/8 ma
62	WRTMUXSEL0PAD_L	Output	--------	--------	11.64	8/8 ma
63	WRTMUXSEL1PAD_L	Output	--------	--------	11.54	8/8 ma

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
64	VDD	+5V POWER				
65	CAS15PAD_L	Output	--------	--------	10.70	12/24 ma
66	CAS14PAD_L	Output	--------	--------	10.72	12/24 ma
67	CAS13PAD_L	Output	--------	--------	10.66	12/24 ma
68	CAS12PAD_L	Output	--------	--------	10.78	12/24 ma
69	CAS11PAD_L	Output	--------	--------	10.76	12/24 ma
70	VSS	GROUND				
71	CAS10PAD_L	Output	--------	--------	10.75	12/24 ma
72	CAS09PAD_L	Output	--------	--------	10.76	12/24 ma
73	CAS08PAD_L	Output	--------	--------	10.76	12/24 ma
74	CAS07PAD_L	Output	--------	--------	10.75	12/24 ma
75	CAS06PAD_L	Output	--------	--------	10.80	12/24 ma
76	VDD	+5V POWER				
77	CAS05PAD_L	Output	--------	--------	10.76	12/24 ma
78	CAS04PAD_L	Output	--------	--------	10.87	12/24 ma
79	CAS03PAD_L	Output	--------	--------	10.80	12/24 ma
80	CAS02PAD_L	Output	--------	--------	10.80	12/24 ma
81	CAS01PAD_L	Output	--------	--------	10.80	12/24 ma
82	VSS	GROUND				
83	CAS00PAD_L	Output	--------	--------	10.80	12/24 ma
84	TESTMODEPAD_L	Input	0.00	0.00	--------	--------
85	MUXADR00PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
86	MUXADR01PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
87	MUXADR02PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
88	VSS	GROUND				
89	MUXADR03PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
90	MUXADR04PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
91	MUXADR05PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
92	MUXADR06PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
93	MUXADR07PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
94	VDD	+5V POWER				
95	MUXADR08PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
96	MUXADR09PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
97	MUXADR10PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
98	MUXADR11PAD_H	Bidi	0.00	0.00	See Note 3	12/24 ma
99	VSS	GROUND				
100	TSPAD_L	Input	5.84	1.55	--------	--------
101	VDD	+5V POWER				
102	SCANTESTPAD_L	Input	5.26	0.00	--------	--------
103	SCANCLKPAD_H	Input	Scan Test Clock			
104	VSS	GROUND				
105	VSS	GROUND				
106	TBSTPAD_L	Input	5.32	1.24	--------	--------
107	TSIZ2PAD_H	Input	3.33	1.07	--------	--------
108	TSIZ1PAD_H	Input	2.60	1.11	--------	--------
109	TSIZ0PAD_H	Input	3.36	1.04	--------	--------
110	TT3PAD_H	Input	6.54	1.43	--------	--------
111	VDD	+5V POWER				
112	TT1PAD_H	Input	6.00	1.54	--------	--------
113	AACKPAD_L	Bidi	4.13	1.07	11.62	12/24 ma
114	ARTRYPAD_L	Bidi	4.74	2.08	13.43	12/24 ma
115	TEAPAD_L	Bidi	4.50	1.20	12.92	12/24 ma
116	TAPAD_L	Bidi	4.25	0.83	15.24	12/24 ma
117	VSS	GROUND				
118	SSD2PAD_L	Output	--------	--------	10.90	8/8 ma
119	SSD1PAD_L	Output	--------	--------	10.81	8/8 ma

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
120	SSD0PAD_L	Output	--------	--------	10.88	8/8 ma
121	BR4PAD_L	Input	4.12	1.64	--------	--------
122	BR3PAD_L	Input	4.07	1.36	--------	--------
123	VSS	GROUND				
124	BR2PAD_L	Input	3.25	1.61	--------	--------
125	BR1PAD_L	Input	4.41	1.19	--------	--------
126	BR0PAD_L	Input	4.50	1.58	--------	--------
127	RDDA2PAD_L	Input	3.57	0.00	--------	--------
128	RDDA1PAD_L	Input	2.82	0.01	--------	--------
129	VDD	+5V POWER				
130	RDDA0PAD_L	Input	1.87	0.47	--------	--------
131	SACK2PAD_L	Input	7.58	0.50	--------	--------
132	SACK1PAD_L	Input	7.37	0.43	--------	--------
133	SACK0PAD_L	Input	7.50	0.47	--------	--------
134	DBWOPAD_L	Output	--------	--------	13.18	8/8 ma
135	DBG4PAD_L	Output	--------	--------	12.04	8/8 ma
136	DBG3PAD_L	Output	--------	--------	11.93	8/8 ma
137	DBG2PAD_L	Output	--------	--------	12.17	8/8 ma
138	DBG1PAD_L	Output	--------	--------	12.28	8/8 ma
139	DBG0PAD_L	Output	--------	--------	12.70	8/8 ma
140	VSS	GROUND				
141	BG4PAD_L	Output	--------	--------	14.55	8/8 ma
142	BG3PAD_L	Output	--------	--------	14.43	8/8 ma
143	BG2PAD_L	Output	--------	--------	14.38	8/8 ma
144	BG1PAD_L	Output	--------	--------	14.46	8/8 ma
145	BG0PAD_L	Output	--------	--------	14.43	8/8 ma
146	VSS	GROUND				
147	DT7PAD_H	Bidi	0.00	0.00	12.44	8/8 ma

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
148	DT6PAD_H	Bidi	0.00	0.00	12.39	8/8 ma
149	DT5PAD_H	Bidi	0.00	0.00	12.32	8/8 ma
150	DT4PAD_H	Bidi	0.00	0.00	12.40	8/8 ma
151	VDD	+5V POWER				
152	DT3PAD_H	Bidi	0.00	0.00	12.35	8/8 ma
153	DT2PAD_H	Bidi	0.00	0.00	12.31	8/8 ma
154	DT1PAD_H	Bidi	0.00	0.00	12.30	8/8 ma
155	DT0PAD_H	Bidi	0.00	0.00	12.30	8/8 ma
156	VSS	GROUND				
157	VSS	GROUND				
158	AD31PAD_H	Input	3.22	0.99	--------	--------
159	AD30PAD_H	Input	3.62	1.15	--------	--------
160	AD29PAD_H	Input	3.17	1.03	--------	--------
161	AD28PAD_H	Input	4.53	0.83	--------	--------
162	AD27PAD_H	Input	4.42	0.87	--------	--------
163	VSS	GROUND				
164	AD26PAD_H	Input	3.29	1.02	--------	--------
165	AD25PAD_H	Input	3.35	0.94	--------	--------
166	AD24PAD_H	Input	3.17	0.96	--------	--------
167	AD23PAD_H	Input	3.11	1.03	--------	--------
168	AD22PAD_H	Input	3.38	0.96	--------	--------
169	AD21PAD_H	Input	3.14	0.95	--------	--------
170	VDD	+5V POWER				
171	AD20PAD_H	Input	2.76	1.10	--------	--------
172	AD19PAD_H	Input	3.25	0.99	--------	--------
173	AD18PAD_H	Input	2.95	1.14	--------	--------
174	AD17PAD_H	Input	3.02	1.21	--------	--------
175	AD16PAD_H	Input	2.92	1.15	--------	--------

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
176	VDD	+5V POWER				
177	VDD	+5V POWER				
178	INTPAD_L	Output	--------	--------	14.60	8/8 ma
179	VSS	GROUND				
180	CLKPAD_H	Input	System Clock			
181	VSS	GROUND				
182	ROMDATAWENPAD_L	Output	--------	--------	12.68	12/12 ma
183	ROMDATAOENPAD_L	Output	--------	--------	13.46	12/12 ma
184	ROMADDR28PAD_H	Output	--------	--------	12.67	12/12 ma
185	ROMADDR27PAD_H	Bidi	0.00	0.00	12.36	12/12 ma
186	ROMALEPAD_L	Output	--------	--------	12.27	12/12 ma
187	TAGALEPAD_L	Output	--------	--------	11.97	8/8 ma
188	VSS	GROUND				
189	RESETPAD_L	Input	(Asynchronous)			
190	AD15PAD_H	Input	2.60	1.35	--------	--------
191	AD14PAD_H	Input	2.71	1.34	--------	--------
192	AD13PAD_H	Input	2.76	1.16	--------	--------
193	AD12PAD_H	Input	2.73	1.13	--------	--------
194	AD11PAD_H	Input	2.62	1.29	--------	--------
195	VDD	+5V POWER				
196	AD10PAD_H	Input	2.75	1.29	--------	--------
197	AD09PAD_H	Input	3.65	0.94	--------	--------
198	AD08PAD_H	Input	3.43	0.99	--------	--------
199	AD07PAD_H	Input	4.14	1.64	--------	--------
200	AD06PAD_H	Input	3.92	1.54	--------	--------
201	VSS	GROUND				
202	AD05PAD_H	Input	3.39	1.62	--------	--------
203	AD04PAD_H	Input	3.54	1.41	--------	--------

TABLE 37. Pinout by number

Note: All names with a "PAD_H" suffix are active high; all with a "PAD_L" suffix are active low. All I/O are TTL-level CMOS, with slew-rate-controlled output drivers. Outputs are specified into a lumped 80pF load. All timing values are in nanoseconds, and are measured with respect to CLKPAD_H.

Pin #	Name	Signal Type	Setup Time	Hold Time	Output Time	Output Driver Strength (H/L)
204	AD03PAD_H	Input	3.15	1.37	--------	--------
205	AD02PAD_H	Input	3.13	1.46	--------	--------
206	AD01PAD_H	Input	3.30	1.26	--------	--------
207	AD00PAD_H	Input	3.53	1.71	--------	--------
208	VSS	GROUND				

TABLE 38. Summary of pin usage

Signal type	Quantity
Bidirectionals	43
Inputs	57
Outputs	63
Total Signal Pins	163
Power	15
Ground	30
TOTAL PINS	208

Notes:

1) TAG setup values with respect to CLK are based upon Figure 34 (see Ttagsu), and are shown in Table 39.

FIGURE 34 Tag Input Setup Timing

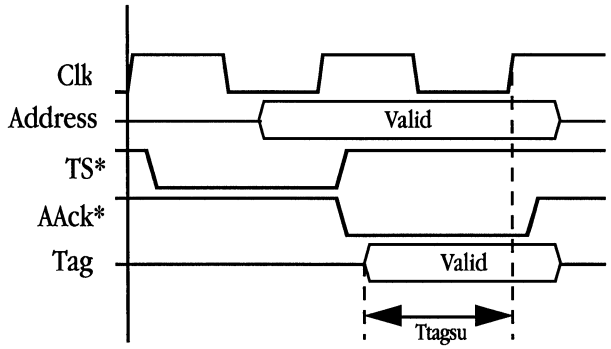


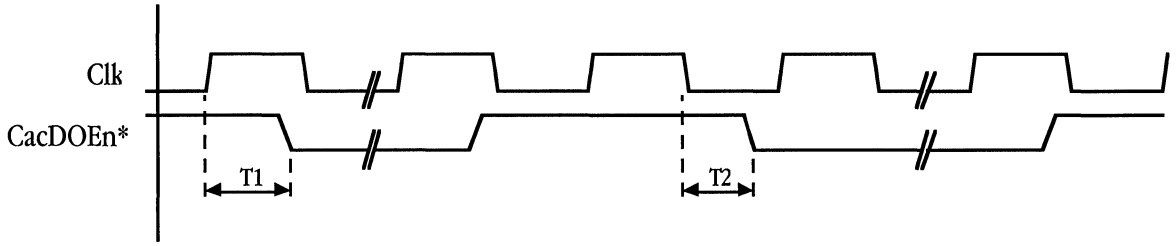
TABLE 39. TAG Pin setup values

Signal Name	Setup (Ttagsu)
TAG00PAD_H	13.03
TAG01PAD_H	12.84
TAG02PAD_H	12.79
TAG03PAD_H	13.08
TAG04PAD_H	13.13
TAG05PAD_H	12.99
TAG06PAD_H	12.85
TAG07PAD_H	12.73
TAG08PAD_H	13.09
TAG09PAD_H	13.09
TAG10PAD_H	13.25
TAG11PAD_H	13.23
TAG12PAD_H	12.97
TAG13PAD_H	12.46
TAGVALIDPAD_H	12.65
TAGDIRTYPAD_H	11.77

2) CACDOENPAD_L output timing has two components: One from the positive edge of CLK, and one from the negative edge. Figure 35 shows the meaning behind the two components.

$$T1 = 12.85 \text{ ns, and } T2 = 8.97 \text{ ns.}$$

FIGURE 35 CacDOEn Output Timing



3) MuxAdr output timing has two components: One (T3) is a combinatorial function of ARBus address, and the other (T4) is from internal Clk->Q. Figure 36 shows the timing measurements, and Table 40 shows the values of those measurements.

FIGURE 36 MuxAdr Output Timing

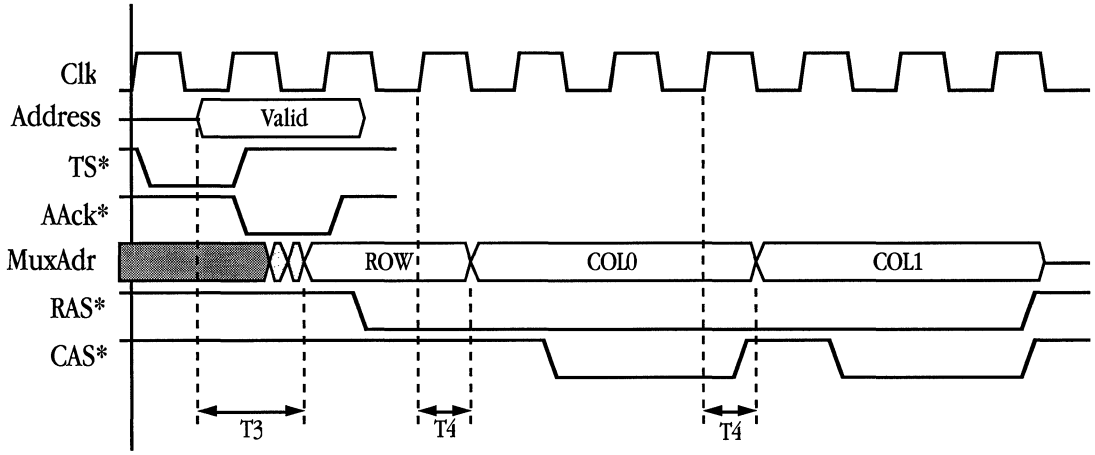


TABLE 40. MuxAdr Output Values

Signal Name	T3	T4
MUXADR00PAD_H	23.61	11.56
MUXADR01PAD_H	23.61	11.54
MUXADR02PAD_H	23.57	11.52

Signal Name	T3	T4
MUXADR03PAD_H	23.65	11.56
MUXADR04PAD_H	23.64	11.60
MUXADR05PAD_H	23.65	11.61
MUXADR06PAD_H	23.61	11.52
MUXADR07PAD_H	23.67	11.60
MUXADR08PAD_H	23.65	11.53
MUXADR09PAD_H	23.65	11.61
MUXADR10PAD_H	23.69	11.61
MUXADR11PAD_H	23.68	11.60