Alpha 21164 Microprocessor

Data Sheet

Order Number: EC–QAEPB–TE

Revision/Update Information: This document supersedes the *Alpha 21164 Microprocessor Data Sheet* EC–QAEPA–TE.

Digital Equipment Corporation
Maynard, Massachusetts

April 1995

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Figures

Tables

1 About This Data Sheet

This data sheet provides a technical overview of the Alpha 21164 microprocessor, including:

- Functional units
- Signal descriptions
- External interface
- Internal processor register (IPR) summary
- Privileged architecture library code (PALcode) instructions
- Electrical characteristics
- Thermal characteristics
- Mechanical packaging

This data sheet is not intended to provide the reader with everything needed to begin chip implementation. For a more comprehensive description of the 21164 and the Alpha architecture, refer to documents listed in the Technical Support and Ordering Information section located at the end of this document.

Document Conventions

Throughout this data sheet, the following conventions are used:

- INT*n* refers to NATURALLY ALIGNED groups of *n* 8-bit bytes. For example:
  - INT16—The four least significant address bits are 0.
  - INT8—The three least significant address bits are 0.
  - INT4—The two least significant address bits are 0.
- Values of 1, 0, and X are used in some tables. The X signifies a *don't care* (1 or 0) convention, which can be determined by the system designer.

2 Alpha 21164 Microprocessor Features

- Fully pipelined 64-bit advanced RISC architecture supports multiple operating systems, including:
  - Microsoft Windows NT
  - OSF/1
  - OpenVMS
- 266-MHz through 300-MHz operation
- Superscalar 4-way instruction issue
- High-bandwidth (128-bit) interface
- Peak execution rate of 1200 MIPS
- 0.50- μm CMOS technology
- Three onchip caches:
  - 8K-byte, direct-mapped, L1 instruction cache
  - 8K-byte, dual-ported, direct-mapped, write-through L1 data cache
  - 96K-byte, 3-way, set-associative, write-back L2 data and instruction cache
- Supports optional board-level L3 cache ranging from 1M byte to 64M bytes

The 21164 microprocessor implements IEEE S_floating and T_floating, and VAX F_floating and G_floating data types and supports longword (32-bit) and quadword (64-bit) integers. Provides byte (8-bit) and word (16-bit) support by byte-manipulation instructions. Limited hardware support is provided for the VAX D_floating data type.

3 Microarchitecture

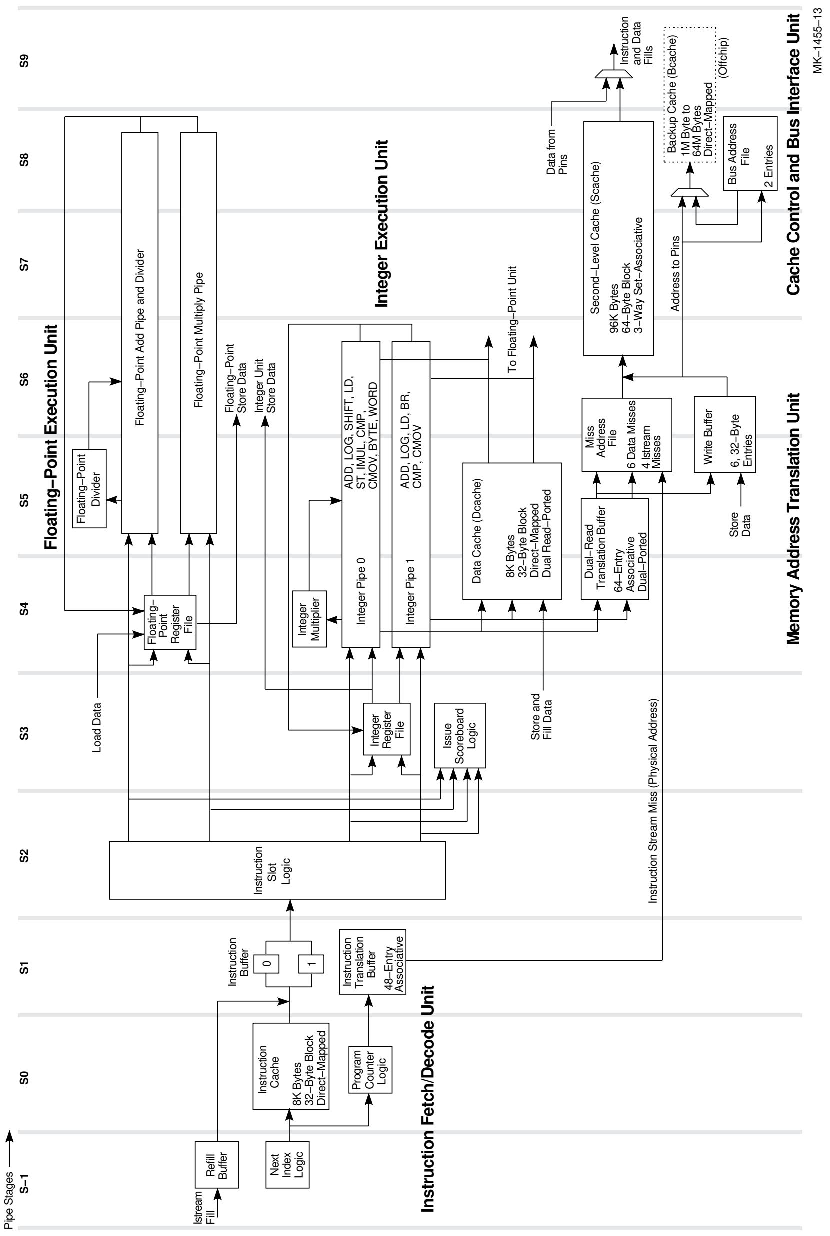
The Alpha 21164 Microprocessor is a high-performance implementation of Digital's Alpha architecture. The following sections provide an overview of the chip's architecture and major functional units.

Figure 1 is a block diagram of the 21164. A larger version of this figure is printed on a foldout page at the end of the *Alpha 21164 Microprocessor Hardware Reference Manual*.

The 21164 consists of the following sections (Figure 1):

- Instruction fetch/decode and branch unit (Ibox)
- Integer execution unit (Ebox)
- Memory address translation unit (Mbox)
- Cache control and bus interface unit (Cbox)
- Floating-point execution unit (Fbox)
- Data cache (Dcache)
- Instruction cache (Icache)
- Secondary cache (Scache)
- Serial read-only memory (SROM) interface

Figure 1 Alpha 21164 Microprocessor Block/Pipe Flow Diagram



3.1 Instruction Fetch/Decode and Branch Unit

The primary function of the instruction fetch/decode and branch unit (Ibox) is to manage and issue instructions to the Ebox, Mbox, and Fbox. It also manages the instruction cache. The Ibox contains:

- Prefetcher and instruction buffer
- Instruction slot and issue logic
- Program counter (PC) and branch prediction logic
- 48-entry instruction translation buffers (ITBs)
- Abort logic
- Register conflict logic
- Interrupt and exception logic

3.1.1 Instruction Prefetch and Decode

The Ibox handles only NATURALLY ALIGNED groups of four instructions (INT16). The Ibox does not advance to a new group of four instructions until all instructions in a group are issued. If a branch to the middle of an INT16 group occurs, then the Ibox attempts to issue the instructions from the branch target to the end of the current INT16, then it proceeds to the next INT16 of instructions after all the instructions in the target INT16 are issued. Thus, proper code scheduling is required to achieve optimal performance.

3.1.2 Branch Prediction

The branch unit, or prediction logic, is also part of the Ibox. Branch and PC prediction are necessary to predict and begin fetching the target instruction stream before the branch or jump instruction is issued. Each instruction location in the instruction cache (Icache) contains a 2-bit history state to record the outcome of branch instructions.

3.1.3 Instruction Translation Buffer

The Ibox includes a 48-entry, fully associative instruction translation buffer (ITB). The buffer stores recently used instruction stream (Istream) address translations and protection information for pages ranging from 8 to 512 kilobytes and uses a not-last-used replacement algorithm.

The 21164 provides two optional translation extensions called superpages. Access to superpages is allowed only while executing in privileged mode.

- One superpage maps virtual address bits <39:13> to physical address bits <39:13>, on a one-to-one basis, when virtual address bits <42:41> equal 2.

- The other superpage maps virtual address bits <29:13> to physical address bits <29:13>, on a one-to-one basis, and forces physical address bits <39:30> to 0 when virtual address bits <42:30> equal 1FFE(hex).

3.1.4 Interrupts

The Ibox exception logic supports three sources of interrupts:

- Hardware interrupts

  There are seven level-sensitive hardware interrupt sources supplied by the following signals:

  irq_h<3:0>
  sys_mch_chk_irq_h
  pwr_fail_irq_h
  mch_halt_irq_h

- Software interrupts

  There are 15 prioritized software interrupts sourced by an onchip internal processor register (IPR).

- Asynchronous system traps

  There are four asynchronous system traps (ASTs) controlled by onchip IPRs.

  Most interrupts can be independently masked in onchip enable registers. In addition, AST interrupts are qualified by the current processor mode. All interrupts are disabled when the processor is executing PALcode.

3.2 Integer Execution Unit

The integer execution unit (Ebox) contains two 64-bit integer execution pipelines—E0 and E1, which include the following:

- Two adders
- Two logic boxes
- A barrel shifter
- Byte-manipulation logic
- An integer multiplier

The Ebox also includes the 40-entry, 64-bit integer register file (IRF) that contains the 32 integer registers defined by the Alpha architecture and 8 PALshadow registers. The register file has four read ports and two write ports, which provide operands to both integer execution pipelines and accept results from both pipes. The register file also accepts load instruction results (memory data) on the same two write ports.

3.3 Floating-Point Execution Unit

The onchip, pipelined floating-point unit (FPU) can execute both IEEE and VAX floating-point instructions. The 21164 supports IEEE S_floating and T_floating data types, and all rounding modes. It also supports VAX F_floating and G_floating data types, and provides limited support for the D_floating format. The FPU contains:

- A 32-entry, 64-bit floating-point register file (FRF).
- A user-accessible control register.
- A floating-point multiply pipeline.
- A floating-point add pipeline—The floating-point divide unit is associated with the floating-point add pipeline but is not pipelined.

The FPU can accept two instructions every cycle, with the exception of floating-point divide instructions. The result latency for nondivide, floating-point instructions is four cycles.

3.4 Memory Address Translation Unit

The memory address translation unit (Mbox) contains three major sections:

- Data translation buffer (dual ported)
- Miss address file (MAF)
- Write buffer address file

The Mbox receives up to two virtual addresses every cycle from the Ebox. The translation buffer generates the corresponding physical addresses and access control information for each virtual address. The 21164 implements a 43-bit virtual address and a 40-bit physical address.

3.4.1 Data Translation Buffer

The 64-entry, fully associative, dual-read-ported data translation buffer (DTB) stores recently used data stream (Dstream) page table entries (PTEs). Each entry supports all four granularity hint-bit combinations, so that a single DTB entry can provide translation for up to 512 contiguously mapped, 8K-byte pages.

The DTB also supports the register-enabled superpage extension. The DTB superpage maps provide virtual-to-physical address translation for two regions of the virtual address space.

3.4.2 Miss Address File

The Mbox begins the execution of each load instruction by translating the virtual address and by accessing the data cache (Dcache). Translation and Dcache tag read operations occur in parallel. If the addressed location is found in the Dcache (a hit), then the data from the Dcache is formatted and written to either the integer register file (IRF) or floating-point register file (FRF). The formatting required depends on the particular load instruction executed. If the data is not found in the Dcache (a miss), then the address, target register number, and formatting information are entered in the miss address file (MAF).

The MAF performs a load-merging function. When a load miss occurs, each MAF entry is checked to see if it contains a load miss that addresses the same Dcache (32-byte) block. If it does, and certain merging rules are satisfied, then the new load miss is merged with an existing MAF entry. This allows the Mbox to service two or more load misses with one data fill from the Cbox.

There are six MAF entries for load misses and four more for Ibox instruction fetches and prefetches. Load misses are usually the highest Mbox priority.

3.4.3 Store Execution

The Dcache follows a write-through protocol. During the execution of a store instruction, the Mbox probes the Dcache to determine whether the location to be overwritten is currently cached. If so (a Dcache hit), the Dcache is updated. Regardless of the Dcache state, the Mbox forwards the data to the Cbox.

A load instruction that is issued one cycle after a store instruction in the pipeline creates a conflict if both the load and store operations access the same memory location. (The store instruction has not yet updated the location when the load instruction reads it.) This conflict is handled by forcing the load instruction to take a replay trap; that is, the Ibox flushes the pipeline and restarts execution from the load instruction. By the time the load instruction arrives at the Dcache the second time, the conflicting store instruction has written the Dcache and the load instruction is executed normally.

Replay traps can be avoided by scheduling the load instruction to issue three cycles after the store instruction. If the load instruction is scheduled to issue two cycles after the store instruction, then it will be issue-stalled for one cycle.

3.4.4 Write Buffer

The Mbox also contains a write buffer that has six 32-byte entries. The write buffer provides a finite, high-bandwidth resource for receiving store data to minimize the number of CPU stall cycles.

3.5 Cache Control and Bus Interface Unit

The cache control and bus interface unit (Cbox) processes all accesses sent by the Mbox and implements all memory-related external interface functions, particularly the coherence protocol functions for write-back caching. It controls the second-level cache (Scache) and the optional board-level backup cache (Bcache). The Cbox handles all instruction and primary Dcache read misses, performs the function of writing data from the write buffer into the shared coherent memory subsystem, and has a major role in executing the Alpha memory barrier (MB) instruction. The Cbox also controls the 128-bit bidirectional data bus, address bus, and I/O control.

3.6 Cache Organization

The 21164 has three onchip caches—a primary L1 data cache, a primary L1 instruction cache, and a second-level L2 combined data and instruction cache. All memory cells in the onchip caches are fully static, 6-transistor, CMOS structures.

The 21164 also provides control for an optional board-level, external L3 cache.

3.6.1 Data Cache

The data cache (Dcache) is a dual-read-ported, single-write-ported, 8K-byte cache. It is a write-through, read-allocate, direct-mapped, physical cache with 32-byte blocks.

3.6.2 Instruction Cache

The instruction cache (Icache) is an 8K-byte, virtual, direct-mapped cache with 32-byte blocks. Each block tag contains:

- A 7-bit address space number (ASN) field as defined by the Alpha architecture
- A 1-bit address space match (ASM) field as defined by the Alpha architecture
- A 1-bit PALcode (physically addressed) indicator

Software, rather than Icache hardware, maintains Icache coherence with memory.

3.6.3 Second-Level Cache

The second-level cache (Scache) is a 96K-byte, 3-way, set-associative, physical, write-back, write-allocate cache with 32- or 64-byte blocks. It is a mixed data and instruction cache. The Scache is fully pipelined; it processes read and write operations at the rate of one INT16 per CPU cycle and can alternate between read and write accesses without bubble cycles.

When operating in 32-byte block mode, the Scache has 64-byte blocks with 32-byte subblocks, one tag per block. If configured to 32 bytes, the Scache is organized as three sets of 512 blocks, with each block divided into two 32-byte subblocks. If configured to 64 bytes, the Scache is three sets of 512 64-byte blocks.

3.6.4 External Cache

The Cbox implements control for an optional, external, direct-mapped, physical, write-back, write-allocate cache with 32- or 64-byte blocks. The 21164 supports board-level cache sizes of 1, 2, 4, 8, 16, 32, and 64 megabytes.

3.7 Serial Read-Only Memory Interface

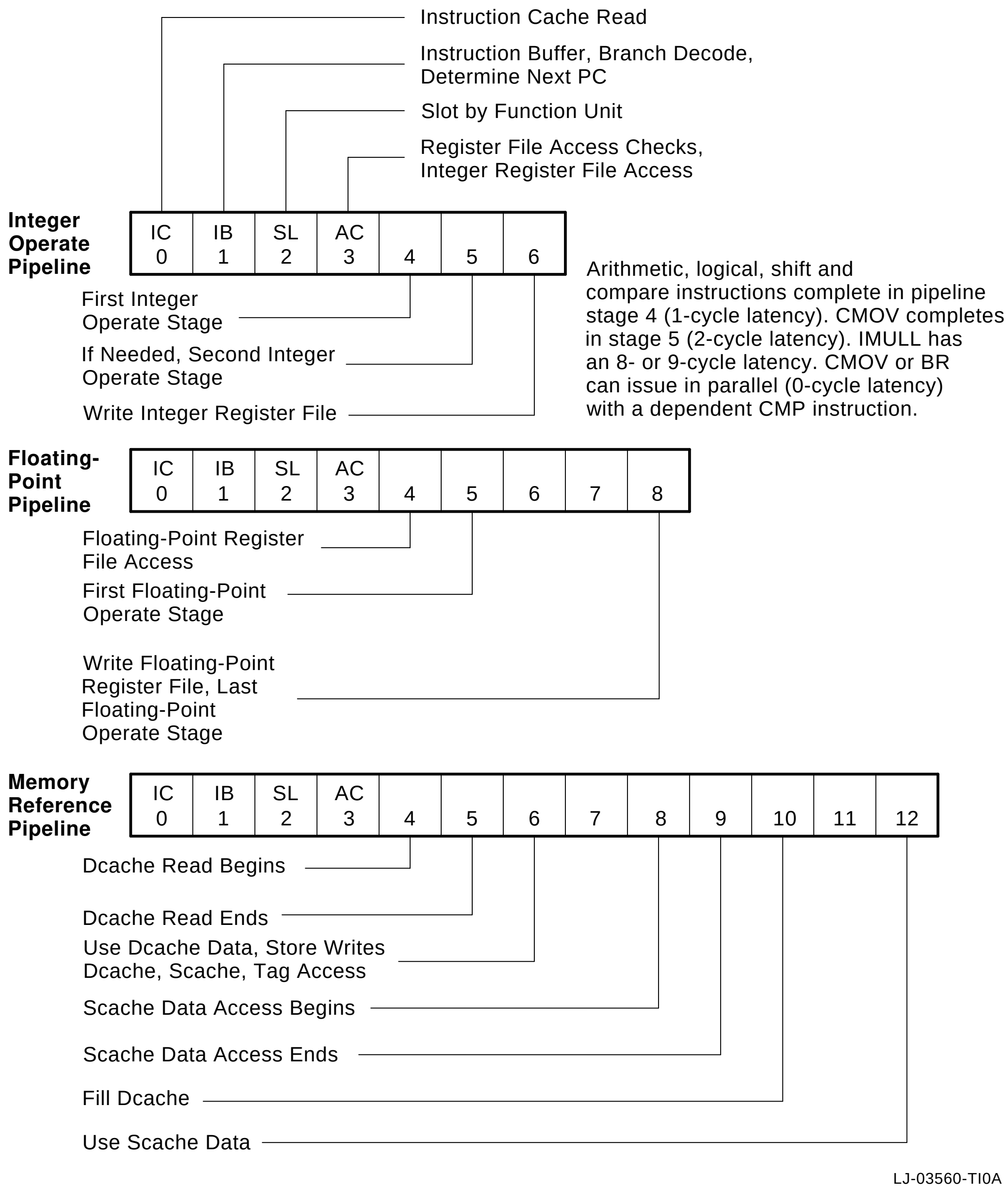
The serial read-only memory (SROM) interface provides the initialization data load path from a system SROM to the instruction cache. Following initialization, this interface can function as a diagnostic port by using privileged architecture library code (PALcode).

3.8 Pipeline Organization

The 21164 has a 7-stage (or 7-cycle) pipeline for integer operate and memory reference instructions, and a 9-stage pipeline for floating-point operate instructions. The Ibox maintains state for all pipeline stages to track outstanding register write operations.

Figure 2 shows the integer operate, memory reference, and floating-point operate pipelines for the Ibox, FPU, Ebox, and Mbox. The first four stages are executed in the Ibox. Remaining stages are executed by the Ebox, Fbox, Mbox, and Cbox.

Figure 2 Instruction Pipeline Stages



4 Pinout and Signal Descriptions

Sections 4.1 and 4.2 list and describe the 21164 microprocessor external signals, and their associated pins.

4.1 Pin Assignment

The 21164 package has 499 pins aligned in an interstitial pin grid array (IPGA) design. Table 1 lists the 21164 signal pins and their corresponding pin grid array (PGA) locations in alphabetic order. There are 292 functional signal pins, 2 spare (unused) signal pins, 104 power (**Vdd**) pins, and 101 ground (**Vss**) pins.

Table 1 Alphabetic Signal Pin List

Signal	PGA Location	Signal	PGA Location	Signal	PGA Location
addr_bus_req_h	E23	**addr_cmd_par_h**	B20	**addr_h<4>**	BB14
addr_h<5>	BC13	**addr_h<6>**	BA13	**addr_h<7>**	AV14
addr_h<8>	AW13	**addr_h<9>**	BC11	**addr_h<10>**	BA11
addr_h<11>	AV12	**addr_h<12>**	AW11	**addr_h<13>**	BC09
addr_h<14>	BA09	**addr_h<15>**	AV10	**addr_h<16>**	AW09
addr_h<17>	BC07	**addr_h<18>**	BA07	**addr_h<19>**	AV08
addr_h<20>	AW07	**addr_h<21>**	BC05	**addr_h<22>**	BC39
addr_h<23>	AW37	**addr_h<24>**	AV36	**addr_h<25>**	BA37
addr_h<26>	BC37	**addr_h<27>**	AW35	**addr_h<28>**	AV34
addr_h<29>	BA35	**addr_h<30>**	BC35	**addr_h<31>**	AW33
addr_h<32>	AV32	**addr_h<33>**	BA33	**addr_h<34>**	BC33
addr_h<35>	AW31	**addr_h<36>**	AV30	**addr_h<37>**	BA31
addr_h<38>	BC31	**addr_h<39>**	BB30	**addr_res_h<0>**	C27
addr_res_h<1>	F26	**addr_res_h<2>**	E27	**cack_h**	G21
cfail_h	C25	**clk_mode_h<0>**	AU21	**clk_mode_h<1>**	BA23
cmd_h<0>	F20	**cmd_h<1>**	A19	**cmd_h<2>**	C19
cmd_h<3>	E19	**cpu_clk_out_h**	BA25	**dack_h**	B24
data_bus_req_h	E25	**data_check_h<0>**	J41	**data_check_h<1>**	K38
data_check_h<2>	J39	**data_check_h<3>**	G43	**data_check_h<4>**	G41

(continued on next page)

Table 1 (Cont.) Alphabetic Signal Pin List

Signal	PGA Location	Signal	PGA Location	Signal	PGA Location
data_check_h<5>	H38	**data_check_h<6>**	G39	**data_check_h<7>**	E43
data_check_h<8>	J03	**data_check_h<9>**	K06	**data_check_h<10>**	J05
data_check_h<11>	G01	**data_check_h<12>**	G03	**data_check_h<13>**	H06
data_check_h<14>	G05	**data_check_h<15>**	E01	**data_h<0>**	J43
data_h<1>	L39	**data_h<2>**	M38	**data_h<3>**	L41
data_h<4>	L43	**data_h<5>**	N39	**data_h<6>**	P38
data_h<7>	N41	**data_h<8>**	N43	**data_h<9>**	P42
data_h<10>	R39	**data_h<11>**	T38	**data_h<12>**	R41
data_h<13>	R43	**data_h<14>**	U39	**data_h<15>**	V38
data_h<16>	U41	**data_h<17>**	U43	**data_h<18>**	W39
data_h<19>	W41	**data_h<20>**	W43	**data_h<21>**	Y38
data_h<22>	Y42	**data_h<23>**	AA39	**data_h<24>**	AA41
data_h<25>	AA43	**data_h<26>**	AB38	**data_h<27>**	AC43
data_h<28>	AC41	**data_h<29>**	AC39	**data_h<30>**	AD42
data_h<31>	AD38	**data_h<32>**	AE43	**data_h<33>**	AE41
data_h<34>	AE39	**data_h<35>**	AG43	**data_h<36>**	AG41
data_h<37>	AF38	**data_h<38>**	AG39	**data_h<39>**	AJ43
data_h<40>	AJ41	**data_h<41>**	AH38	**data_h<42>**	AJ39
data_h<43>	AK42	**data_h<44>**	AL43	**data_h<45>**	AL41
data_h<46>	AK38	**data_h<47>**	AL39	**data_h<48>**	AN43
data_h<49>	AN41	**data_h<50>**	AM38	**data_h<51>**	AN39
data_h<52>	AR43	**data_h<53>**	AR41	**data_h<54>**	AP38
data_h<55>	AR39	**data_h<56>**	AU43	**data_h<57>**	AU41
data_h<58>	AT38	**data_h<59>**	AU39	**data_h<60>**	AW43
data_h<61>	AW41	**data_h<62>**	AV38	**data_h<63>**	AW39
data_h<64>	J01	**data_h<65>**	L05	**data_h<66>**	M06
data_h<67>	L03	**data_h<68>**	L01	**data_h<69>**	N05
data_h<70>	P06	**data_h<71>**	N03	**data_h<72>**	N01

(continued on next page)

Table 1 (Cont.) Alphabetic Signal Pin List

Signal	PGA Location	Signal	PGA Location	Signal	PGA Location
data_h<73>	P02	**data_h<74>**	R05	**data_h<75>**	T06
data_h<76>	R03	**data_h<77>**	R01	**data_h<78>**	U05
data_h<79>	V06	**data_h<80>**	U03	**data_h<81>**	U01
data_h<82>	W05	**data_h<83>**	W03	**data_h<84>**	W01
data_h<85>	Y06	**data_h<86>**	Y02	**data_h<87>**	AA05
data_h<88>	AA03	**data_h<89>**	AA01	**data_h<90>**	AB06
data_h<91>	AC01	**data_h<92>**	AC03	**data_h<93>**	AC05
data_h<94>	AD02	**data_h<95>**	AD06	**data_h<96>**	AE01
data_h<97>	AE03	**data_h<98>**	AE05	**data_h<99>**	AG01
data_h<100>	AG03	**data_h<101>**	AF06	**data_h<102>**	AG05
data_h<103>	AJ01	**data_h<104>**	AJ03	**data_h<105>**	AH06
data_h<106>	AJ05	**data_h<107>**	AK02	**data_h<108>**	AL01
data_h<109>	AL03	**data_h<110>**	AK06	**data_h<111>**	AL05
data_h<112>	AN01	**data_h<113>**	AN03	**data_h<114>**	AM06
data_h<115>	AN05	**data_h<116>**	AR01	**data_h<117>**	AR03
data_h<118>	AP06	**data_h<119>**	AR05	**data_h<120>**	AU01
data_h<121>	AU03	**data_h<122>**	AT06	**data_h<123>**	AU05
data_h<124>	AW01	**data_h<125>**	AW03	**data_h<126>**	AV06
data_h<127>	AW05	**data_ram_oe_h**	F22	**data_ram_we_h**	A23
dc_ok_h	AU23	**fill_error_h**	A25	**fill_h**	G23
fill_id_h	F24	**fill_nocheck_h**	G25	**idle_bc_h**	A27
index_h<4>	A29	**index_h<5>**	C29	**index_h<6>**	F28
index_h<7>	E29	**index_h<8>**	B30	**index_h<9>**	A31
index_h<10>	C31	**index_h<11>**	F30	**index_h<12>**	E31
index_h<13>	A33	**index_h<14>**	C33	**index_h<15>**	F32
index_h<16>	E33	**index_h<17>**	A35	**index_h<18>**	C35
index_h<19>	F34	**index_h<20>**	E35	**index_h<21>**	A37
index_h<22>	C37	**index_h<23>**	F36	**index_h<24>**	E37

(continued on next page)

Table 1 (Cont.) Alphabetic Signal Pin List

Signal	PGA Location	Signal	PGA Location	Signal	PGA Location
index_h<25>	A39	**int4_valid_h<0>**	F38	**int4_valid_h<1>**	E41
int4_valid_h<2>	F06	**int4_valid_h<3>**	E03	**irq_h<0>**	BA29
irq_h<1>	AU27	**irq_h<2>**	BC29	**irq_h<3>**	AW27
mch_hlt_irq_h	AU25	**osc_clk_in_h**	BC21	**osc_clk_in_l**	BB22
perf_mon_h	AW29	**port_mode_h<0>**	AY20	**port_mode_h<1>**	BB20
pwr_fail_irq_h	AV26	**ref_clk_in_h**	AW25	**scache_set_h<0>**	C17
scache_set_h<1>	A17	**shared_h**	C23	**srom_clk_h**	BA19
srom_data_h	BC19	**srom_oe_l**	AW19	**srom_present_l**	AV20
st_clk_h	E05	**system_lock_flag_h**	G27	**sys_clk_out1_h**	AW23
sys_clk_out1_l	BB24	**sys_clk_out2_h**	AV24	**sys_clk_out2_l**	BC25
sys_mch_chk_irq_h	BA27	**sys_reset_l**	BC27	**tag_ctl_par_h**	F18
tag_data_h<20>	A05	**tag_data_h<21>**	E07	**tag_data_h<22>**	F08
tag_data_h<23>	C07	**tag_data_h<24>**	A07	**tag_data_h<25>**	E09
tag_data_h<26>	F10	**tag_data_h<27>**	C09	**tag_data_h<28>**	A09
tag_data_h<29>	E11	**tag_data_h<30>**	F12	**tag_data_h<31>**	C11
tag_data_h<32>	A11	**tag_data_h<33>**	E13	**tag_data_h<34>**	F14
tag_data_h<35>	C13	**tag_data_h<36>**	A13	**tag_data_h<37>**	B14
tag_data_h<38>	E15	**tag_data_par_h**	C15	**tag_dirty_h**	E17
tag_ram_oe_h	C21	**tag_ram_we_h**	A21	**tag_shared_h**	A15
tag_valid_h	F16	**tck_h**	AW17	**tdi_h**	BC17
tdo_h	BA17	**temp_sense**	AW15	**test_status_h<0>**	BA15
test_status_h<1>	AV16	**tms_h**	AV18	**trst_l**	BC15
victim_pending_h	E21	**spare_in<438>**	E39	**spare_io<250>**	AV28

(continued on next page)

Table 1 (Cont.) Alphabetic Signal Pin List

Signal	PGA Location
Vss—Metal planes 2[1] and 5[2]	A03, A41, AA07, AA37, AC07, AC37, AD04, AD40, AF02, AF42, AG07, AG37, AH04, AH40, AL07, AL37, AM04, AM40, AP02, AP42, AR07, AR37, AT04, AT40, AU09, AU13, AU17, AU31, AU35, AV02, AV22, AV42, AW21, AY04, AY08, AY12, AY16, AY22, AY24, AY28, AY32, AY36, AY40, B02, B06, B10, B18, B26, B34, B38, B42, BA01, BA21, BA43, BB02, BB06, BB10, BB18, BB26, BB34, BB38, BB42, BC03, BC41, C01, C43, D04, D08, D12, D16, D20, D24, D28, D32, D36, D40, F02, F42, G09, G13, G17, G31, G35, H04, H40, J07, J37, K02, K42, M04, M40, N07, N37, T04, T40, U07, U37, V02, V42, Y04, Y40
Vdd Metal planes 4 and 6	AB02, AB04, AB40, AB42, AE07, AE37, AF04, AF40, AH02, AH42, AJ07, AJ37, AK04, AK40, AM02, AM42, AN07, AN37, AP04, AP40, AT02, AT42, AU07, AU11, AU15, AU19, AU29, AU33, AU37, AV04, AV40, AY02, AY06, AY10, AY14, AY18, AY26, AY30, AY34, AY38, AY42, B04, B08, B12, B16, B22, B28, B32, B36, B40, BA03, BA05, BA39, BA41, BB04, BB08, BB12, BB16, BB28, BB32, BB36, BB40, BC23, C03, C05, C39, C41, D02, D06, D10, D14, D18, D22, D26, D30, D34, D38, D42, F04, F40, G11, G15, G19, G29, G33, G37, H02, H42, K04, K40, L07, L37, M02, M42, P04, P40, R07, R37, T02, T42, V04, V40, W07, W37

[1]Metal plane 2—Seal ring connection tied to **Vss**

[2]Metal plane 5—Heat slug braze pad connections tied to **Vss**

4.2 Alpha 21164 Packaging

Figure 3 shows the 21164 pinout from the top view with pins facing down.

Figure 3 Alpha 21164 Top View (Pin Down)

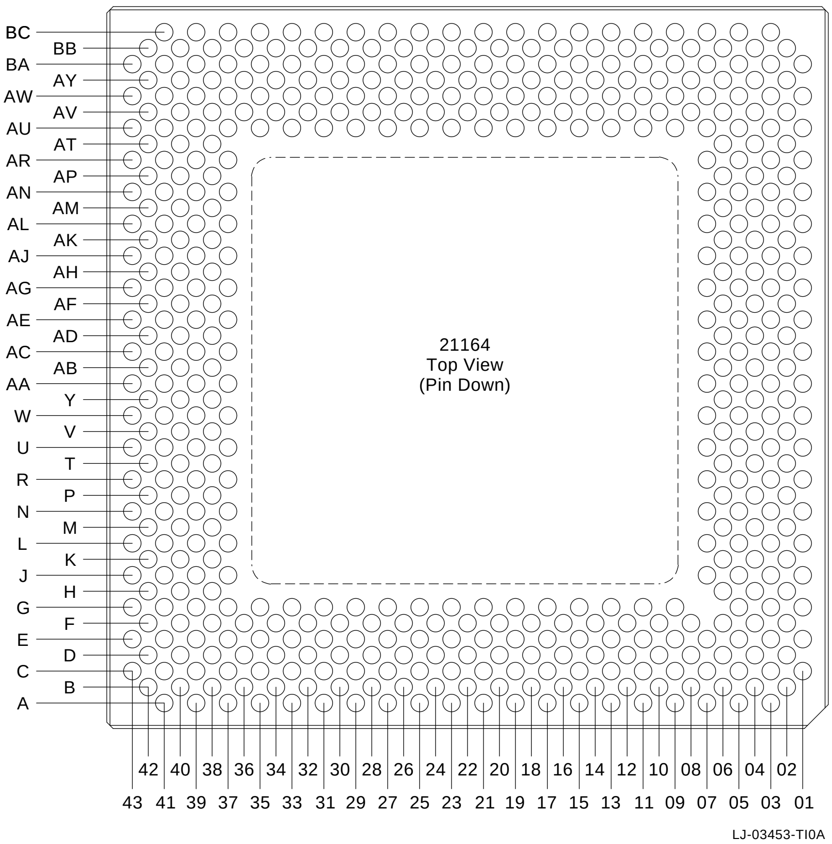
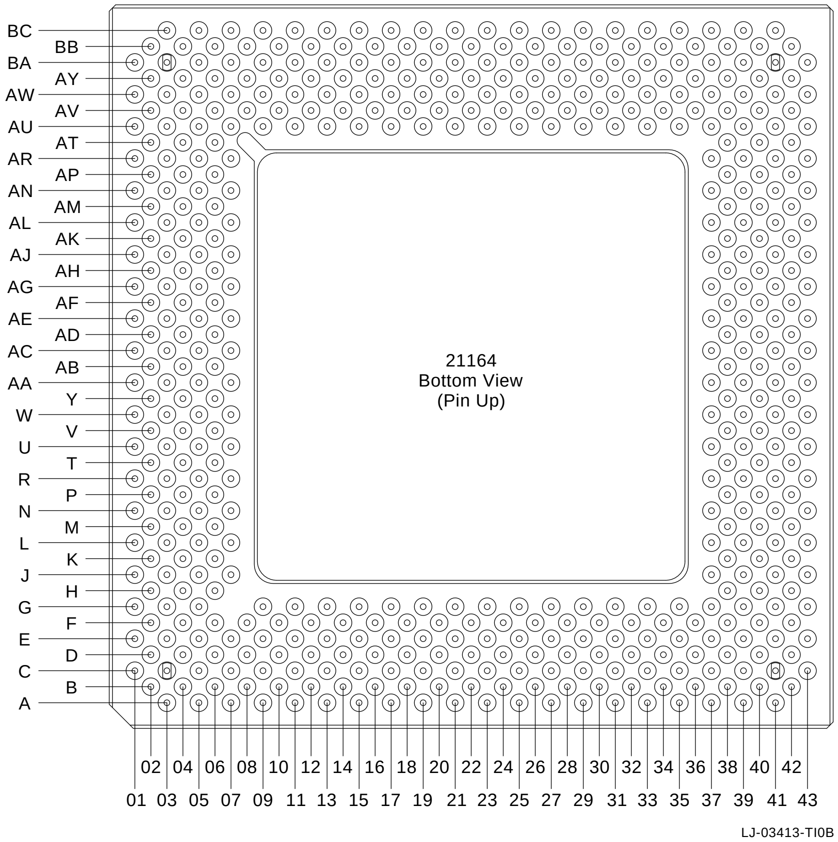


Figure 4 shows the 21164 pinout from the bottom view with pins facing up.

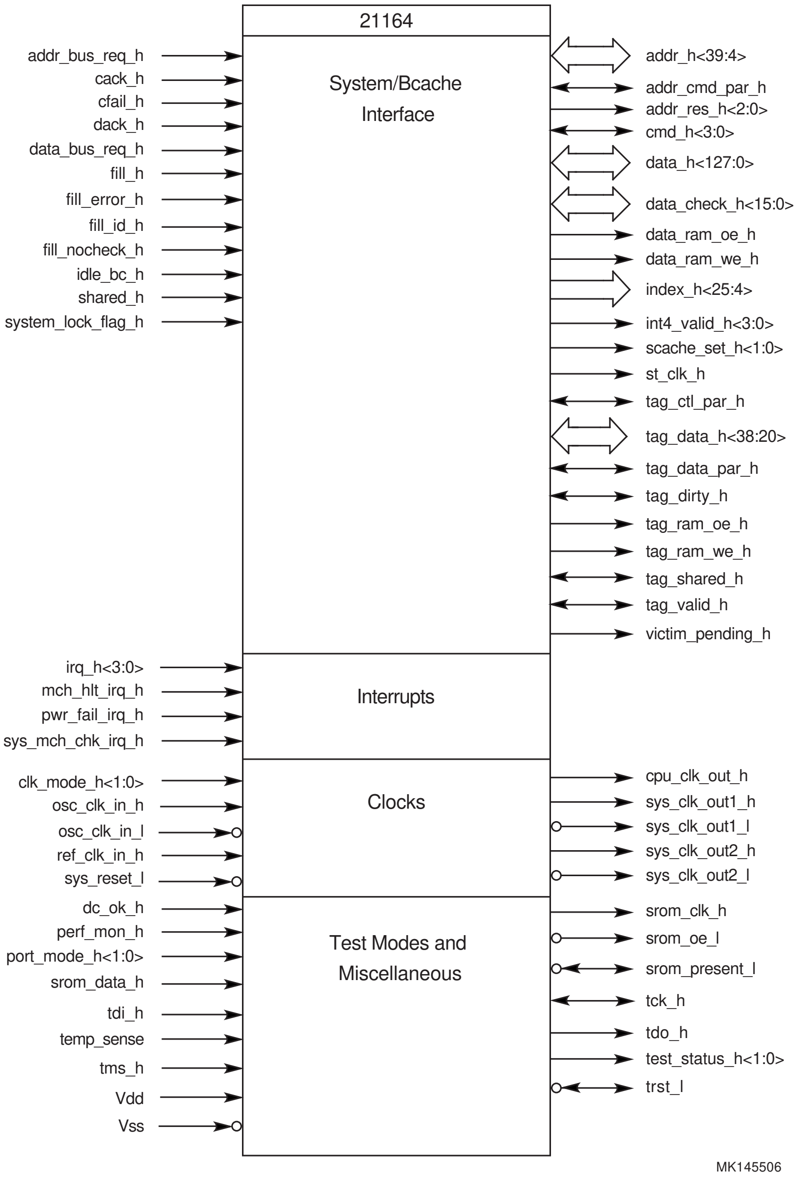
Figure 4 Alpha 21164 Bottom View (Pin Up)



4.3 Alpha 21164 Microprocessor Logic Symbol

Figure 5 shows the logic symbol for the 21164 chip.

Figure 5 Alpha 21164 Microprocessor Logic Symbol



4.4 Alpha 21164 Signal Names and Functions

The following table defines the 21164 signal types referred to in this section:

Signal Type	Definition
B	Bidirectional
I	Input only
O	Output only

The remaining two tables describe the function of each 21164 external signal. Table 2 lists all signals in alphanumeric order. This table provides full signal descriptions. Table 3 lists signals by function and provides an abbreviated description.

Table 2 Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
addr_h<39:4>	B	36	Address bus. These bidirectional signals provide the address of the requested data or operation between the 21164 and the system. If bit 39 is asserted, then the reference is to noncached, I/O memory space.
addr_bus_req_h	I	1	Address bus request. The system interface uses this signal to gain control of the **addr_h<39:4>**, **addr_cmd_par_h**, and **cmd_h<3:0>** pins.
addr_cmd_par_h	B	1	Address command parity. This is the odd parity bit on the current command and address buses. The 21164 takes a machine check if a parity error is detected. The system should do the same if it detects an error.
addr_res_h<1:0>	O	2	Address response bits <1> and <0>. For system commands, the 21164 uses these pins to indicate the state of the block in the Scache:

Bits	Command	Meaning
00	NOP	Nothing.
01	NOACK	Data not found or clean.
10	ACK/Scache	Data from Scache.
11	ACK/Bcache	Data from Bcache.

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
addr_res_h<2>	O	1	Address response bit <2>. For system commands, the 21164 uses this pin to indicate if the command hits in the Scache or onchip load lock register.
cack_h	I	1	Command acknowledge. The system interface uses this signal to acknowledge any one of the commands driven by the 21164.
cfail_h	I	1	Command fail. This signal has two uses. It can be asserted during a cack cycle of a WRITE BLOCK LOCK command to indicate that the write operation is not successful. In this case, both **cack_h** and **cfail_h** are asserted together. It can also be asserted instead of **cack_h** to force an instruction fetch/decode unit (Ibox) timeout event. This causes the 21164 to do a partial reset and trap to the machine check (MCHK) PALcode entry point, which indicates a serious hardware error.
clk_mode_h<1:0>	I	2	Clock test mode. These signals specify a relationship between **osc_clk_in_h,l** and the CPU cycle time. These signals should be deasserted in normal operation mode.
cmd_h<3:0>	B	4	Command bus. These signals drive and receive the commands from the command bus. The following tables define the commands that can be driven on the **cmd_h<3:0>** bus by the 21164 or the system.

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
			21164 Commands to System:

cmd_h <3:0>	Command	Meaning
0000	NOP	Nothing.
0001	LOCK	Lock register address.
0010	FETCH	The 21164 passes a FETCH instruction to the system.
0011	FETCH_M	The 21164 passes a FETCH_M instruction to the system.
0100	MEMORY BARRIER	MB instruction.
0101	SET DIRTY	Dirty bit set if shared bit is clear.
0110	WRITE BLOCK	Request to write a block.
0111	WRITE BLOCK LOCK	Request to write a block with lock.
1000	READ MISS0	Request for data.
1001	READ MISS1	Request for data.
1010	READ MISS MOD0	Request for data; modify intent.
1011	READ MISS MOD1	Request for data; modify intent.
1100	BCACHE VICTIM	Bcache victim should be removed.
1101	—	Reserved.
1110	READ MISS MOD STC0	Request for data, ST*x*_C data.
1111	READ MISS MOD STC1	Request for data, ST*x*_C data.

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
			System Commands to 21164:

cmd_h <3:0>	Command	Meaning
0000	NOP	Nothing.
0001	FLUSH	Remove block from caches; return dirty data.
0010	INVALIDATE	Invalidate the block from caches.
0011	SET SHARED	Block goes to the shared state.
0100	READ	Read a block.
0101	READ DIRTY	Read a block; set shared.
0111	READ DIRTY/INV	Read a block; invalidate.

Signal	Type	Count	Description
cpu_clk_out_h	O	1	CPU clock output. This signal is used for test purposes.
dack_h	I	1	Data acknowledge. The system interface uses this signal to control data transfer between the 21164 and the system.
data_h<127:0>	B	128	Data bus. These signals are used to move data between the 21164, the system, and the Bcache.
data_bus_req_h	I	1	Data bus request. If the 21164 samples this signal asserted on the rising edge of sysclk *n*, then the 21164 does not drive the data bus on the rising edge of sysclk *n*+1. Before asserting this signal, the system should assert **idle_bc_h** for the correct number of cycles. If the 21164 samples this signal deasserted on the rising edge of sysclk *n*, then the 21164 drives the data bus on the rising edge of sysclk *n*+1.
data_check_h<15:0>	B	16	Data check. These signals set even byte parity or INT8 ECC for the current data cycle.

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
data_ram_oe_h	O	1	Data RAM output enable. This signal is asserted for Bcache read operations.
data_ram_we_h	O	1	Data RAM write-enable. This signal is asserted for any Bcache write operation.
dc_ok_h	I	1	dc voltage OK. Must be deasserted until dc voltage reaches proper operating level. After that, **dc_ok_h** is asserted.
fill_h	I	1	Fill warning. If the 21164 samples this signal asserted on the rising edge of sysclk *n*, then the 21164 provides the address indicated by **fill_id_h** to the Bcache on the rising edge of sysclk *n*+1. The Bcache begins to write in that sysclk. At the end of sysclk *n*+1, the 21164 waits for the next sysclk and then begins the write operation again if **dack_h** is not asserted.
fill_error_h	I	1	Fill error. If this signal is asserted during a fill from memory, it indicates to the 21164 that the system has detected an invalid address or hard error. The system still provides an apparently normal read sequence with correct ECC/parity though the data is not valid. The 21164 traps to the machine check (MCHK) PALcode entry point and indicates a serious hardware error. **fill_error_h** should be asserted when the data is returned. Each assertion produces a MCHK trap.
fill_id_h	I	1	Fill identification. Asserted with **fill_h** to indicate which register is used. The 21164 supports two outstanding load instructions. If this signal is asserted when the 21164 samples **fill_h** asserted, then the 21164 provides the address from miss register 1. If it is deasserted, then the address in miss register 0 is used for the read operation.
fill_nocheck_h	I	1	Fill checking off. If this signal is asserted, then the 21164 does not check the parity or ECC for the current data cycle on a fill.
idle_bc_h	I	1	Idle Bcache. When asserted, the 21164 finishes the current Bcache read or write operation but does not start a new read or write operation until the signal is deasserted. The system interface must assert this signal in time to idle the Bcache before fill data arrives.
index_h<25:4>	O	22	Index. These signals index the Bcache.

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
int4_valid_h<3:0>	O	4	INT4 data valid. During write operations to noncached space, these signals are used to indicate which INT4 bytes of data are valid. This is useful for noncached write operations that have been merged in the write buffer.

int4_valid_h<3:0>	Write Meaning
*xxx*1	**data_h<31:0>** valid
*xx*1*x*	**data_h<63:32>** valid
*x*1*xx*	**data_h<95:64>** valid
1*xxx*	**data_h<127:96>** valid

During read operations to noncached space, these signals indicate which INT8 bytes of a 32-byte block need to be read and returned to the processor. This is useful for read operations to noncached memory.

int4_valid_h<3:0>	Read Meaning
*xxx*1	**data_h<63:0>** valid
*xx*1*x*	**data_h<127:64>** valid
*x*1*xx*	**data_h<191:128>** valid
1*xxx*	**data_h<255:192>** valid

Note: For both read and write operations, multiple **int4_valid_h<3:0>** bits can be set simultaneously.

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
irq_h<3:0>	I	4	System interrupt requests. These signals have multiple modes of operation. During normal operation, these level-sensitive signals are used to signal interrupt requests. During initialization, these signals are used to set up the CPU cycle time divisor for **sys_clk_out1_h,l** as follows:

irq_h <3>	irq_h <2>	irq_h <1>	irq_h <0>	Ratio
Low	Low	High	High	3
Low	High	Low	Low	4
Low	High	Low	High	5
Low	High	High	Low	6
Low	High	High	High	7
High	Low	Low	Low	8
High	Low	Low	High	9
High	Low	High	Low	10
High	Low	High	High	11
High	High	Low	Low	12
High	High	Low	High	13
High	High	High	Low	14
High	High	High	High	15

Signal	Type	Count	Description
mch_hlt_irq_h	I	1	Machine halt interrupt request. This signal has multiple modes of operation. During initialization, this signal is used to set up **sys_clk_out2_h,l** delay. During normal operation, it is used to signal a halt request.
osc_clk_in_h **osc_clk_in_l**	I I	1 1	Oscillator clock inputs. These signals provide the differential clock input that is the fundamental timing of the 21164. These signals are driven at twice the desired internal clock frequency. (Under normal operating conditions the CPU cycle time is one-half the frequency of **osc_clk_in**.)

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
perf_mon_h	I	1	Performance monitor. This signal can be used as an input to the 21164 internal performance monitoring hardware from offchip events (such as bus activity).
port_mode_h<1:0>	I	2	Select test port interface modes (normal, manufacturing, and debug). For normal operation, both signals must be deasserted.
pwr_fail_irq_h	I	1	Power failure interrupt request. This signal has multiple modes of operation. During initialization, this signal is used to set up **sys_clk_out2_h,l** delay. During normal operation, this signal is used to signal a power failure.
ref_clk_in_h	I	1	Reference clock input. Optional. Used to synchronize the timing of multiple microprocessors to a single reference clock. If this signal is not used, it must be tied to **Vdd** for proper operation.
scache_set_h<1:0>	O	2	Secondary cache set. During a read miss request, these signals indicate the Scache set number that will be filled when the data is returned. This information can be used by the system to maintain a duplicate copy of the Scache tag store.
shared_h	I	1	Keep block status shared. For systems without a Bcache, when a WRITE BLOCK/NO VICTIM PENDING or WRITE BLOCK LOCK command is acknowledged, this pin can be used to keep the block status shared or private in the Scache.
srom_clk_h	O	1	Serial ROM clock. Supplies the clock that causes the SROM to advance to the next bit. The cycle time of this clock is 128 times the cycle time of the CPU clock.
srom_data_h	I	1	Serial ROM data. Input for the SROM.
srom_oe_l	O	1	Serial ROM output enable. Supplies the output enable to the SROM.
srom_present_l[1]	B	1	Serial ROM present. Indicates that SROM is present and ready to load the Icache.

[1]This signal is shown as bidirectional. However, for normal operation it is input only. The output function is used during manufacturing test and verification only.

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
st_clk_h	O	1	STRAM clock. Clock for Bcache synchronously timed RAMs (STRAMs). This signal is synchronous with **index_h<25:4>** during private read and write operations, and with **sys_clk_out1_h,l** during read and fill operations.
sys_clk_out1_h **sys_clk_out1_l**	O O	1 1	System clock outputs. Programmable system clock (**cpu_clk_out_h** divided by a value of 3 to 15) is used for board-level cache and system logic.
sys_clk_out2_h **sys_clk_out2_l**	O O	1 1	System clock outputs. A version of **sys_clk_out1_h,l** delayed by a programmable amount from 0 to 7 CPU cycles.
sys_mch_chk_irq_h	I	1	System machine check interrupt request. This signal has multiple modes of operation. During initialization, it is used to set up **sys_clk_out2_h,l** delay. During normal operation, it is used to signal a machine interrupt check request.
sys_reset_l	I	1	System reset. This signal protects the 21164 from damage during initial power-up. It must be asserted until **dc_ok_h** is asserted. After that, it is deasserted and the 21164 begins its reset sequence.
system_lock_flag_h	I	1	System lock flag. During fills, the 21164 logically ANDs the value of the system copy with its own copy to produce the true value of the lock flag.
tag_ctl_par_h	B	1	Tag control parity. This signal indicates odd parity for **tag_valid_h**, **tag_shared_h**, and **tag_dirty_h**. During fills, the system should drive the correct parity based on the state of the valid, shared, and dirty bits.
tag_data_h<38:20>	B	19	Bcache tag data bits. This bit range supports 1M-byte to 64M-byte Bcaches.
tag_data_par_h	B	1	Tag data parity bit. This signal indicates odd parity for **tag_data_h<38:20>**.
tag_dirty_h	B	1	Tag dirty state bit. During fills, the system should assert this signal if the 21164 request is a READ MISS MOD, and the shared bit is not asserted.
tag_ram_oe_h	O	1	Tag RAM output enable. This signal is asserted during any Bcache read operation.

(continued on next page)

Table 2 (Cont.) Alpha 21164 Signal Descriptions

Signal	Type	Count	Description
tag_ram_we_h	O	1	Tag RAM write-enable. This signal is asserted during any tag write operation. During the first CPU cycle of a write operation, the write pulse is deasserted. In the second and following CPU cycles of a write operation, the write pulse is asserted if the corresponding bit in the write pulse register is asserted. Bits **BC_WE_CTL<8:0>** control the shape of the pulse.
tag_shared_h	B	1	Tag shared bit. During fills, the system should drive this signal with the correct value to mark the cache block as shared.
tag_valid_h	B	1	Tag valid bit. During fills, this signal is asserted to indicate that the block has valid data.
tck_h	B	1	JTAG boundary scan clock.
tdi_h	I	1	JTAG serial boundary scan data-in signal.
tdo_h	O	1	JTAG serial boundary scan data-out signal.
temp_sense	I	1	Temperature sense. This signal is used to measure the die temperature and is for manufacturing use only. For normal operation, this signal must be left disconnected.
test_status_h<1:0>	O	2	Icache test status. These signals are used for manufacturing test purposes only to extract Icache test status information from the chip. **test_status_h<0>** is asserted if ICSR<39> is true, on Ibox timeout, or remains asserted if the Icache built-in self-test (BiSt) fails. Also, **test_status_h<0>** outputs the value written by PALcode to **test_status_h<1>** through IPR access.
tms_h	I	1	JTAG test mode select signal.
trst_l[1]	B	1	JTAG test access port (TAP) reset signal.
victim_pending_h	O	1	Victim pending. When asserted, this signal indicates that the current read miss has generated a victim.

[1]This signal is shown as bidirectional. However, for normal operation it is input only. The output function is used during manufacturing test and verification only.

Table 3 lists signals by function and provides an abbreviated description.

Table 3 Alpha 21164 Signal Descriptions by Function

Signal	Type	Count	Description
Clocks			
clk_mode_h<1:0>	I	2	Clock test mode.
cpu_clk_out_h	O	1	CPU clock output.
osc_clk_in_h,l	I	2	Oscillator clock inputs.
ref_clk_in_h	I	1	Reference clock input.
st_clk_h	O	1	Bcache STRAM clock output.
sys_clk_out1_h,l	O	2	System clock outputs.
sys_clk_out2_h,l	O	2	System clock outputs.
sys_reset_l	I	1	System reset.
Bcache			
data_h<127:0>	B	128	Data bus.
data_check_h<15:0>	B	16	Data check.
data_ram_oe_h	O	1	Data RAM output enable.
data_ram_we_h	O	1	Data RAM write-enable.
index_h<25:4>	O	22	Index.
tag_ctl_par_h	B	1	Tag control parity.
tag_data_h<38:20>	B	19	Bcache tag data bits.
tag_data_par_h	B	1	Tag data parity bit.
tag_dirty_h	B	1	Tag dirty state bit.
tag_ram_oe_h	O	1	Tag RAM output enable.
tag_ram_we_h	O	1	Tag RAM write-enable.
tag_shared_h	B	1	Tag shared bit.
tag_valid_h	B	1	Tag valid bit.

(continued on next page)

Table 3 (Cont.) Alpha 21164 Signal Descriptions by Function

Signal	Type	Count	Description
System Interface			
addr_h<39:4>	B	36	Address bus.
addr_bus_req_h	I	1	Address bus request.
addr_cmd_par_h	B	1	Address command parity.
addr_res_h<2:0>	O	3	Address response.
cack_h	I	1	Command acknowledge.
cfail_h	I	1	Command fail.
cmd_h<3:0>	B	4	Command bus.
dack_h	I	1	Data acknowledge.
data_bus_req_h	I	1	Data bus request.
fill_h	I	1	Fill warning.
fill_error_h	I	1	Fill error.
fill_id_h	I	1	Fill identification.
fill_nocheck_h	I	1	Fill checking off.
idle_bc_h	I	1	Idle Bcache.
int4_valid_h<3:0>	O	4	INT4 data valid.
scache_set_h<1:0>	O	2	Secondary cache set.
shared_h	I	1	Keep block status shared.
system_lock_flag_h	I	1	System lock flag.
victim_pending_h	O	1	Victim pending.
Interrupts			
irq_h<3:0>	I	4	System interrupt requests.
mch_hlt_irq_h	I	1	Machine halt interrupt request.
pwr_fail_irq_h	I	1	Power failure interrupt request.
sys_mch_chk_irq_h	I	1	System machine check interrupt request.

(continued on next page)

Table 3 (Cont.) Alpha 21164 Signal Descriptions by Function

Signal	Type	Count	Description
Test Modes and Miscellaneous			
dc_ok_h	I	1	dc voltage OK.
perf_mon_h	I	1	Performance monitor.
port_mode_h<1:0>	I	2	Select test port interface modes (normal, manufacturing, and debug).
srom_clk_h	O	1	Serial ROM clock.
srom_data_h	I	1	Serial ROM data.
srom_oe_l	O	1	Serial ROM output enable.
srom_present_l[1]	B	1	Serial ROM present.
tck_h	B	1	JTAG boundary scan clock.
tdi_h	I	1	JTAG serial boundary scan data in.
tdo_h	O	1	JTAG serial boundary scan data out.
temp_sense	I	1	Temperature sense.
test_status_h<1:0>	O	2	Icache test status.
tms_h	I	1	JTAG test mode select.
trst_l[1]	B	1	JTAG test access port (TAP) reset.

[1]This signal is shown as bidirectional. However, for normal operation it is input only. The output function is used during manufacturing test and verification only.

5 Alpha 21164 Microprocessor Functional Overview

This section provides an overview of 21164 external signals that support the following:

- Clocks
- Bcache interface
- System interface
- Interrupts
- Test modes

See Figure 1 for a block diagram of the 21164.

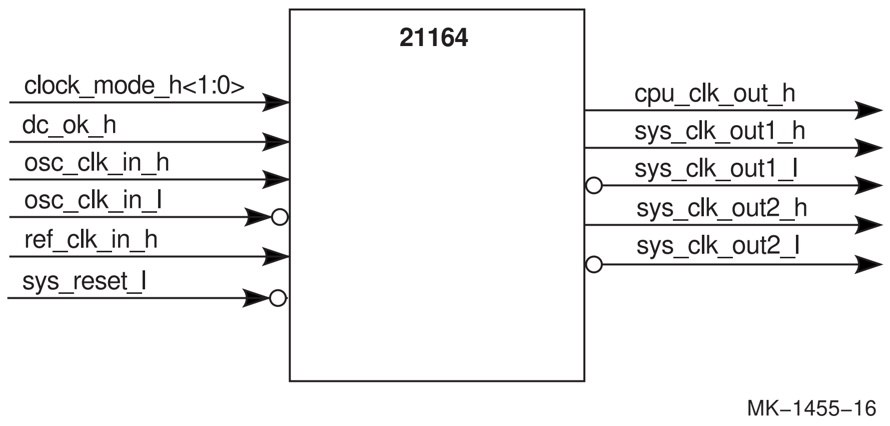
5.1 Clocks

The 21164 accepts two clock signal inputs and develops three clock signal outputs:

Signal	Description
Input Clock Signals	
osc_clk_in_h,l	Differential inputs normally driven at two times the desired internal frequency.
ref_clk_in_h	A system-supplied clock to which the 21164 synchronizes its timing for multiprocessor systems.
Output Clock Signals	
cpu_clk_out_h	A 21164 internal clock that may or may not drive the system clock.
sys_clk_out1_h,l	A clock of programmable speed supplied to the external interface.
sys_clk_out2_h,l	A delayed copy of **sys_clk_out1_h,l**. The delay is programmable and is an integer number of **cpu_clk_out_h** periods.

Figure 6 shows the 21164 clock signals.

Figure 6 Alpha 21164 Clock Signals



5.1.1 CPU Clock

The 21164 uses the differential input clock lines **osc_clk_in_h,l** as a source to generate its CPU clock. The input signals **clk_mode_h<1:0>** control generation of the CPU clock.

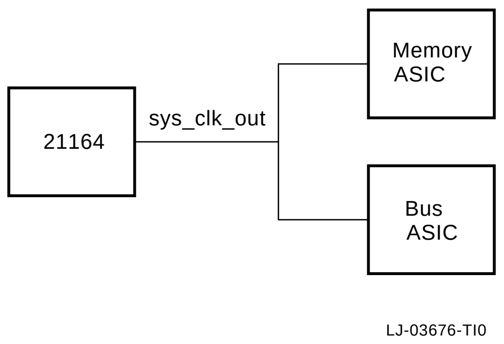
5.1.2 System Clock

The CPU clock is divided by a programmable value of between 3 and 15 to generate a system clock. The programmable feature allows the system designer maximum flexibility when choosing external logic to interface with the 21164.

The **sys_clk_out1_h,l** signals are delayed by a programmable number of CPU cycles between 0 and 7 to produce **sys_clk_out2_h,l**. The output of the programmable divider is symmetric if the divisor is even. The output is asymmetric if the divisor is odd.

Figure 7 shows the 21164 driving the system clock on a uniprocessor system.

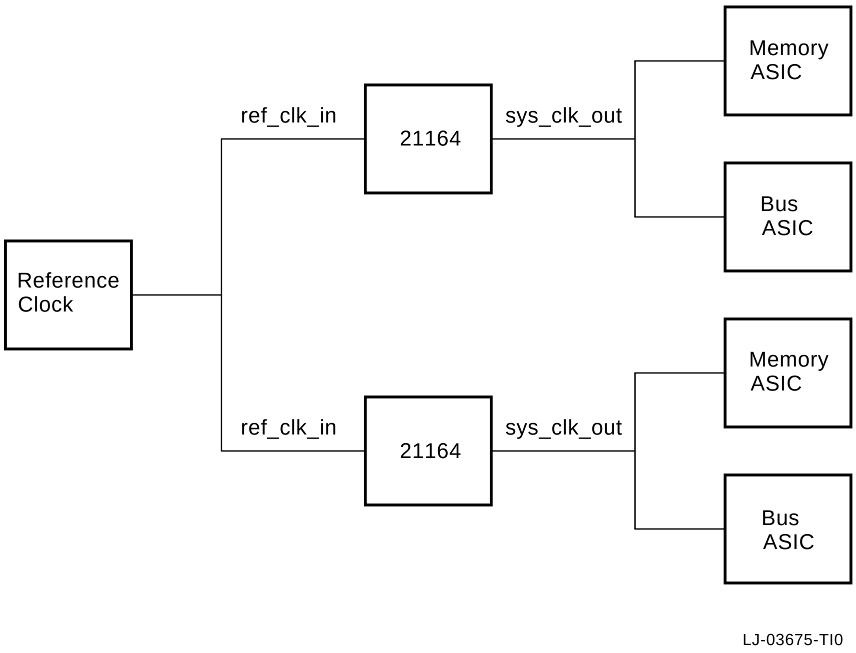
Figure 7 Alpha 21164 Uniprocessor Clock



5.1.3 Reference Clock

The 21164 provides a reference clock input so that other CPUs and system devices can be synchronized in multiprocessor systems. If a clock is asserted on signal **ref_clk_in_h**, then the **sys_clk_out1_h,l** signals are synchronized to that reference clock by means of a digital phase-locked loop (DPLL). Figure 8 shows the 21164 synchronized to a system reference clock.

Figure 8 Alpha 21164 Reference Clock for Multiprocessor Systems



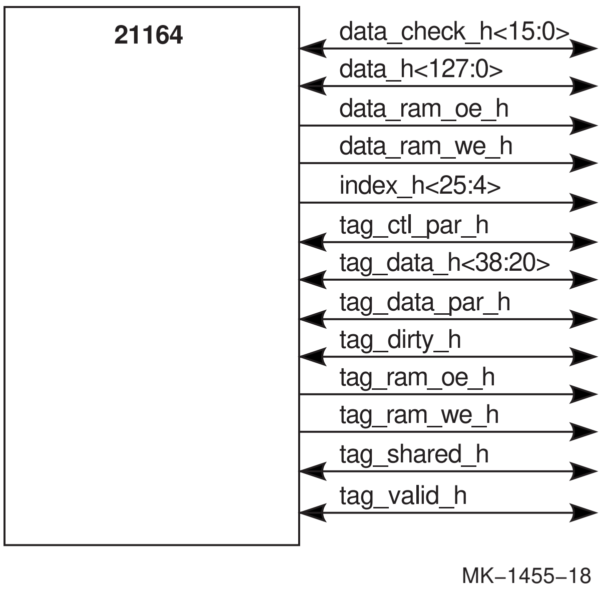
5.2 Board-Level Backup Cache Interface

The 21164 includes an interface and control for an optional board-level backup cache (Bcache). This section describes the Bcache interface. The Bcache interface is made up of the following:

- A data bus (which it shares with the system interface)
- Tag and tag control bits for determining hit and coherence
- SRAM output and SRAM write control signals

Figure 9 shows the 21164 system interface signals.

Figure 9 Alpha 21164 Bcache Interface Signals



The Bcache interface is managed by the cache control and bus interface unit (Cbox). The Bcache interface is a 128-bit bidirectional data bus. The read and write speed of the Bcache can be programmed independently of each other and independently of the system clock ratio. Optionally, the Bcache can operate in a psuedo-pipeline manner. Internal processor registers are used to program the Bcache timing and to enable wave pipelining. See the *Alpha 21164 Microprocessor Hardware Reference Manual* for more information.

The Bcache system supports block sizes of 32 or 64 bytes but it be must set like the secondary cache (Scache). The block size is selected by a mode bit. The Scache is 3-way, set-associative but is a subset of the larger externally implemented, direct-mapped Bcache. In systems with no Bcache, the Scache block size must be set to 64 bytes.

5.2.1 Bcache Victim Buffers

The 21164 is designed to support systems with one or more offchip Bcache victim buffers. External victim buffers improve the overall performance of the Bcache. A Bcache victim is generated when the 21164 deallocates a dirty block from the Bcache. Each time a Bcache victim is produced, the 21164 stops reading the Bcache until the system takes the current victim, and then the Bcache operations resume.

5.2.2 Cache Coherence Protocol

Cache coherency is a concern for single and multiprocessor 21164-based systems as there may be several caches on a processor module and several more in multiprocessor systems.

The system hardware designer need not be concerned about Icache and Dcache coherency. Coherency of the Icache is a software concern—it is flushed with an IMB (PALcode) instruction. The 21164 maintains coherency between the Dcache and the Scache.

If the system does not have a Bcache, the system designer must create mechanisms in the system interface logic to support cache coherency between the Scache, main memory, and other caches in the system.

If the system has a Bcache, the 21164 maintains cache coherency between the Scache and the Bcache. The Scache is a subset of the Bcache. In this case, the designer must create mechanisms in the system interface logic to support cache coherency between the Bcache, main memory, and other caches in the system.

The following tasks must be performed to maintain cache coherency:

- The Cbox in the 21164 maintains coherency in the Dcache and keeps it as a subset of the Scache.
- If an optional Bcache is present, then the 21164 maintains the Scache as a subset of the Bcache. The Scache is set-associative but is kept a subset of the larger externally implemented direct-mapped Bcache.
- System logic must help the 21164 to keep the Bcache coherent with main memory and other caches in the system.
- The Icache is not a subset of any cache and also is not kept coherent with the memory system.

Table 4 describes the Bcache states that determine cache coherence protocol for 21164 systems.

Table 4 Bcache States for Cache Coherency Protocols

Valid[1]	Shared[1]	Dirty[1]	State of Cache Line
0	X	X	Not valid.
1	0	0	Valid for read or write operations. This cache line contains the only cached copy of the block and the copy in memory is identical to this line.
1	0	1	Valid for read or write operations. This cache line contains the only cached copy of the block. The contents of the block have been modified more recently than the copy in memory.
1	1	0	Valid for read or write operations. This block may be in another CPU's cache.
1	1	1	Valid for read or write operations. This block may be in another CPU's cache. The contents of the block have been modified more recently than the copy in memory.

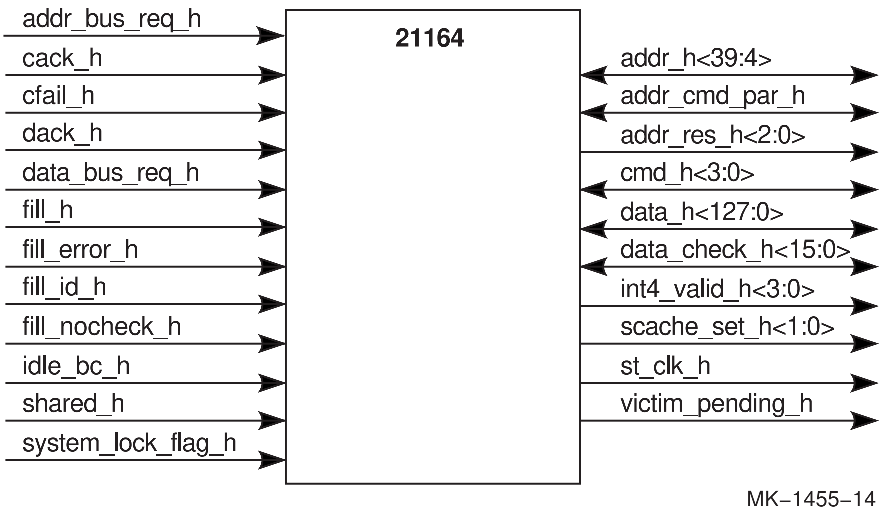
[1] The **tag_valid_h**, **tag_shared_h**, and **tag_dirty_h** signals are described in Table 2.

5.3 System Interface

The system interface is made up of bidirectional address and command buses, a data bus that it shares with the Bcache interface, and several control signals.

Figure 10 shows the 21164 system interface signals.

Figure 10 Alpha 21164 System Interface Signals



The system interface is under the control of the cache control and bus interface unit (Cbox). The system interface is a 128-bit bidirectional data bus. The cycle time of the system interface is programmable to speeds of one-third to one-fifteenth the CPU cycle time. All system interface signals are driven or sampled by the 21164 on the rising edge of **sys_clk_out1_h**.

5.3.1 Commands and Addresses

The 21164 can take up to two commands from the system at a time. The bus interface buffer can hold one or two misses and one or two Scache victim addresses at a time. A miss occurs when the 21164 searches its caches but does not find the addressed block. The 21164 can queue two misses to the system. An Scache victim occurs when the 21164 deallocates a dirty block from the Scache.

The system requests the misses, and the victims arbitrate for the Bcache.

- The highest priority for the Bcache is data movement for the system, which includes fill, read dirty data, invalidate, and set shared activities.

- If there are no system requests for the Bcache, then a 21164 command is selected.

Tables 5 and 6 provide a brief description of the commands that the 21164 and the system can drive on the command bus.

Table 5 Alpha 21164 Commands for the System

cmd<3:0>	Command	Meaning
0000	NOP	Nothing.
0001	LOCK	New lock register address.
0010	FETCH	21164 passes a FETCH to system.
0011	FETCH_M	21164 passes a FETCH_M to system.
0100	MEMORY BARRIER	MB instruction.
0101	SET DIRTY	Dirty bit set if shared bit is clear.
0110	WRITE BLOCK	Request to write a block.
0111	WRITE BLOCK LOCK	Request to write a block with lock.
1000	READ MISS0	Request for data.
1001	READ MISS1	Request for data.
1010	READ MISS MOD0	Request for data; modify intent.
1011	READ MISS MOD1	Request for data; modify intent.
1100	BCACHE VICTIM	Bcache victim should be removed.
1101	—	Spare.
1110	READ MISS MOD STC0	Request for data, ST*x*_C data.
1111	READ MISS MOD STC1	Request for data, ST*x*_C data.

Table 6 System Commands for the 21164

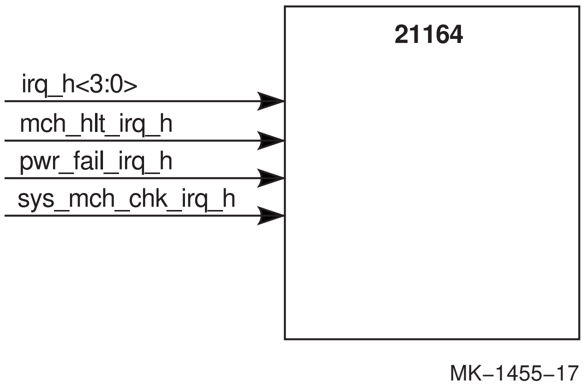
cmd<3:0>	Command	Meaning
0000	NOP	Nothing.
0001	FLUSH	Remove block from caches; return dirty data (flush protocol).
0010	INVALIDATE	Remove the block (write invalidate protocol).
0011	SET SHARED	Block goes to the shared state (write invalidate protocol).
0100	READ	Read a block (flush protocol).
0101	READ DIRTY	Read a block; set shared (write invalidate protocol).
0111	READ DIRTY/INV	Read a block; invalidate (write invalidate protocol).

5.4 Interrupts

The 21164 has seven interrupt signals that have different uses during initialization and normal operation.

Figure 11 shows the 21164 interrupt signals.

Figure 11 Alpha 21164 Interrupt Signals



5.4.1 Interrupt Signals During Initialization

The 21164 interrupt signals work in tandem with the **sys_reset_l** signal to set the values for many of the user-selectable clocking ratios and interface timing parameters. During initialization, the 21164 reads system clock configuration parameters from the interrupt pins.

Table 7 shows the system clock divisor settings. The system clock frequency is determined by dividing the ratio into the CPU clock frequency.

Table 7 System Clock Divisor

irq_h<3>	irq_h<2>	irq_h<1>	irq_h<0>	Ratio
Low	Low	High	High	3
Low	High	Low	Low	4
Low	High	Low	High	5
Low	High	High	Low	6
Low	High	High	High	7
High	Low	Low	Low	8
High	Low	Low	High	9
High	Low	High	Low	10
High	High	High	High	15

Table 8 shows how the three remaining interrupt signals are used to determine the length of the **sys_clk_out2** delay. These signals provide flexible timing for system use.

Table 8 System Clock Delay

sys_mch_chk_irq_h	pwr_fail_irq_h	mch_halt_irq_h	Delay Cycles
Low	Low	Low	0
Low	Low	High	1
Low	High	Low	2
Low	High	High	3
High	Low	Low	4
High	Low	High	5
High	High	Low	6
High	High	High	7

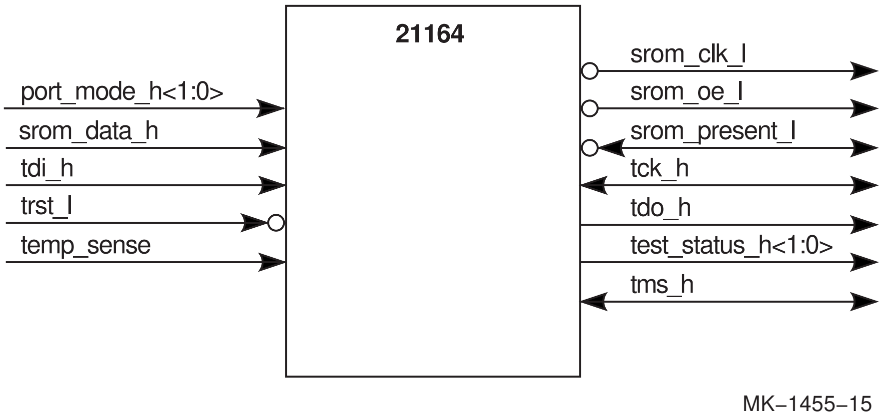
5.4.2 Interrupt Signals During Normal Operation

During normal operation, interrupt signals request various interrupts as described in Table 2.

5.5 Test Modes

Figure 12 shows the 21164 test signals.

Figure 12 Alpha 21164 Test Signals



The 21164 test interface port consists of 13 dedicated signals. Table 9 summarizes the 21164 test port signals and their function.

Table 9 Alpha 21164 Test Port Pins

Pin Name	Type	Function
port_mode_h<1>	I	Must be false.
port_mode_h<0>	I	Must be false.
srom_present_l	I	Tied low if serial ROMs (SROMs) are present in system.
srom_data_h/Rx	I	Receives SROM or serial terminal data.
srom_clk_h/Tx	O	Supplies clock to SROMs or transmits serial terminal data.
srom_oe_l	O	SROM enable.
tdi_h	I	IEEE 1149.1 TDI port.
tdo_h	O	IEEE 1149.1 TDO port.
tms_h	I	IEEE 1149.1 TMS port.
tck_h	B	IEEE 1149.1 TCK port.
trst_l	I	IEEE 1149.1 optional TRST port.
test_status_h<0>	O	Indicates Icache BiSt status.
test_status_h<1>	O	Outputs an IPR-written value and timeout reset.

5.5.1 Normal Test Interface Mode

The test port is in the default or normal test interface mode when the **port_mode_h<1:0>** signals are tied to 00. In this mode, the test port supports the following:

- Serial ROM interface port
- Serial diagnostic terminal interface port
- IEEE 1149.1 test access port

5.5.2 Serial ROM Interface Port

The following signals make up the serial ROM (SROM) interface:

srom_present_l
srom_data_h
srom_oe_l
srom_clk_h

During system reset, the 21164 samples the **srom_present_l** signal for the presence of SROM. If no SROMs are detected at reset, then **srom_present_l** is deasserted and the SROM load is disabled. The reset sequence clears the Icache valid bits, which causes the first instruction fetch to miss the Icache and seek instructions from offchip memory.

If SROMs are present during setup, then the system performs an SROM load as follows:

1. The **srom_oe_l** signal supplies the output enable to the SROM.
2. The **srom_clk_h** signal supplies the clock to the ROM that causes it to advance to the next bit. The cycle time of this clock is 126 ± times the system clock ratio.
3. The **srom_data_h** signal reads the SROM data.

5.5.3 Serial Terminal Port

After the serial ROM data is loaded into the Icache, the three SROM load signals become parallel I/O pins that can drive a diagnostic terminal such as an RS422.

5.5.4 IEEE 1149.1 Test Access Port

The test access port complies with all requirements of the IEEE 1149.1 (JTAG) standard. The following signals make up the test access port:

- **tms_h**—Test access port select.
- **trst_l**—Test access port reset.
- **tck_h**—Test access port clock.
- **tdi_h** and **tdo_h**—Input and output for serial boundary scan, die-ID, bypass, and instruction registers.

5.5.5 Test Status Signals

The **test_status_h** signals extract test status information from the chip.

- The **test_status_h<0>** signal indicates when the Icache built-in self-test (BiSt) fails.
- The **test_status_h<1>** signal detects unrepairable Icache by indicating more than two failing Icache rows.

6 Alpha Architecture Basics

This section provides some basic information about the Alpha architecture. For more detailed information about the Alpha architecture, see the *Alpha Architecture Reference Manual*.

6.1 The Architecture

The Alpha architecture is a 64-bit load and store RISC architecture designed with particular emphasis on speed, multiple instruction issue, multiple processors, and software migration from many operating systems.

All registers are 64 bits in length and all operations are performed between 64-bit registers. All instructions are 32 bits in length. Memory operations are either load or store operations. All data manipulation is done between registers.

The Alpha architecture supports the following data types:

- 8-, 16-, 32-, and 64-bit integers
- IEEE 32-bit and 64-bit floating-point formats
- VAX architecture 32-bit and 64-bit floating-point formats

In the Alpha architecture, instructions interact with each other only by one instruction writing to a register or memory location and another instruction reading from that register or memory location. This use of resources makes it easy to build implementations that issue multiple instructions every CPU cycle.

The 21164 uses a set of subroutines, called privileged architecture library code (PALcode), that is specific to a particular Alpha operating system implementation and hardware platform. These subroutines provide operating system primitives for context switching, interrupts, exceptions, and memory management. These subroutines can be invoked by hardware or CALL_PAL instructions. CALL_PAL instructions use the function field of the instruction to vector to a specified subroutine. PALcode is written in standard machine code with some implementation-specific extensions to provide direct access to low-level hardware functions. PALcode supports optimizations for multiple operating systems, flexible memory-management implementations, and multi-instruction atomic sequences.

The Alpha architecture performs byte shifting and masking with normal 64-bit, register-to-register instructions; it does not include single-byte load and store instructions.

6.2 Addressing

The basic addressable unit in the Alpha architecture is the 8-bit byte. The 21164 supports a 43-bit virtual address.

Virtual addresses as seen by the program are translated into physical memory addresses by the memory-management mechanism. The 21164 supports a 40-bit physical address.

6.3 Integer Data Types

Alpha architecture supports four integer data types:

Data Type	Description
Byte	A byte is 8 contiguous bits that start at an addressable byte boundary. A byte is an 8-bit value. A byte is supported in Alpha architecture by the EXTRACT, MASK, INSERT, and ZAP instructions.
Word	A word is 2 contiguous bytes that start at an arbitrary byte boundary. A word is a 16-bit value. A word is supported in Alpha architecture by the EXTRACT, MASK, and INSERT instructions.
Longword	A longword is 4 contiguous bytes that start at an arbitrary byte boundary. A longword is a 32-bit value. A longword is supported in the Alpha architecture by sign-extended load and store instructions and by longword arithmetic instructions.
Quadword	A quadword is 8 contiguous bytes that start at an arbitrary byte boundary. A quadword is supported in Alpha architecture by load and store instructions and quadword integer operate instructions.

Note

Alpha implementations may impose a significant performance penalty when accessing operands that are not NATURALLY ALIGNED. Refer to the *Alpha Architecture Reference Manual* for details.

6.4 Floating-Point Data Types

The 21164 supports the following floating-point data types:

- Longword integer format in floating-point unit
- Quadword integer format in floating-point unit
- IEEE floating-point formats
  - S_floating
  - T_floating
- VAX floating-point formats
  - F_floating
  - G_floating
  - D_floating (limited support)

7 Alpha 21164 Microprocessor IEEE Floating-Point Conformance

The 21164 supports the IEEE floating-point operations as defined by the Alpha architecture. Support for a complete implementation of the IEEE *Standard for Binary Floating-Point Arithmetic* (ANSI/IEEE Standard 754 1985) is provided by a combination of hardware and software as described in the *Alpha Architecture Reference Manual*.

Additional information about writing code to support precise exception handling (necessary for complete conformance to the standard) is in the *Alpha Architecture Reference Manual*.

The following information is specific to the 21164:

- Invalid operation (INV)

  The invalid operation trap is always enabled. If the trap occurs, then the destination register is UNPREDICTABLE. This exception is signaled if any VAX architecture operand is nonfinite (reserved operand or dirty zero) and the operation can take an exception (that is, certain instructions, such as CPYS, never take an exception). This exception is signaled if any IEEE operand is nonfinite (NAN, INF, denorm) and the operation can take an exception. This trap is also signaled for an IEEE format divide of +/– 0 divided by +/– 0. If the exception occurs, then FPCR<INV> is set and the trap is signaled to the Ibox.

- Divide-by-zero (DZE)

  The divide-by-zero trap is always enabled. If the trap occurs, then the destination register is UNPREDICTABLE. For VAX architecture format, this exception is signaled whenever the numerator is valid and the denominator is zero. For IEEE format, this exception is signaled whenever the numerator is valid and non-zero, with a denominator of +/– 0. If the exception occurs, then FPCR<DZE> is set and the trap is signaled to the Ibox.

  For IEEE format divides, 0/0 signals INV, not DZE.

- Floating overflow (OVF)

  The floating overflow trap is always enabled. If the trap occurs, then the destination register is UNPREDICTABLE. The exception is signaled if the rounded result exceeds in magnitude the largest finite number, which can be represented by the destination format. This applies only to operations whose destination is a floating-point data type. If the exception occurs, then FPCR<OVF> is set and the trap is signaled to the Ibox.

- Underflow (UNF)

  The underflow trap can be disabled. If underflow occurs, then the destination register is forced to a true zero, consisting of a full 64 bits of zero. This is done even if the proper IEEE result would have been –0. The exception is signaled if the rounded result is smaller in magnitude than the smallest finite number that can be represented by the destination format. If the exception occurs, then FPCR<UNF> is set. If the trap is enabled, then the trap is signaled to the Ibox. The 21164 never produces a denormal number; underflow occurs instead.

- Inexact (INE)

  The inexact trap can be disabled. The destination register always contains the properly rounded result, whether the trap is enabled. The exception is signaled if the rounded result is different from what would have been produced if infinite precision (infinitely wide data) were available. For floating-point results, this requires both an infinite precision exponent and fraction. For integer results, this requires an infinite precision integer and an integral result. If the exception occurs, then FPCR<INE> is set. If the trap is enabled, then the trap is signaled to the Ibox.

  The IEEE-754 specification allows INE to occur concurrently with either OVF or UNF. Whenever OVF is signaled (if the inexact trap is enabled), INE is also signaled. Whenever UNF is signaled (if the inexact trap is enabled), INE is also signaled. The inexact trap also occurs concurrently with integer overflow. All valid opcodes that enable INE also enable both overflow and underflow.

  If a CVTQL results in an integer overflow (IOV), then FPCR<INE> is automatically set. (The INE trap is never signaled to the Ibox because there is no CVTQL opcode that enables the inexact trap.)

- Integer overflow (IOV)

  The integer overflow trap can be disabled. The destination register always contains the low-order bits (<64> or <32>) of the true result (not the truncated bits). Integer overflow can occur with CVTTQ, CVTGQ, or CVTQL. In conversions from floating to quadword integer or longword integer, an integer overflow occurs if the rounded result is outside the range $-2^{63} .. 2^{63-1}$. In conversions from quadword integer to longword integer, an integer overflow occurs if the result is outside the range $-2^{31} .. 2^{31-1}$. If the exception occurs, then the appropriate bit in the floating-point control register (FPCR) is set. If the trap is enabled, then the trap is signaled to the Ibox.

- Software completion (SWC)

  The software completion signal is not recorded in the FPCR. The state of this signal is always sent to the Ibox. If the Ibox detects the assertion of any of the listed exceptions concurrent with the assertion of the SWC signal, then it sets EXC_SUM<SWC>.

Input exceptions always take priority over output exceptions. If both exception types occur, then only the input exception is recorded in the FPCR and only the input exception is signaled to the Ibox.

8 Internal Processor Registers

The tables in this section provide a summary of the 21164 implementation-specific internal processor registers (IPRs). For detailed register information, see the *Alpha 21164 Microprocessor Hardware Reference Manual*. For more information about the architecturally specified IPRs, see the *Alpha Architecture Reference Manual*.

8.1 Ibox, Mbox, Dcache, and PALtemp IPRs

Table 10 lists the Ibox, Mbox, data cache (Dcache), and PALtemp IPRs. These IPRs are accessible to PALcode by means of the HW_MTPR and HW_MFPR instructions using the IPR index. The Ibox holds a bank of 24 PALtemp registers.

Table 10 Ibox, Mbox, Dcache, and PALtemp Internal Processor Registers

Mnemonic	Register Name	Access	Index (hex)
	Ibox IPRs		
ISR	Interrupt summary	R	100
ITB_TAG	Istream translation buffer tag	W	101
ITB_PTE	Instruction translation buffer page table entry	R/W	102
ITB_ASN	Instruction translation buffer address space number	R/W	103
ITB_PTE_TEMP	Instruction translation buffer page table entry temporary	R	104
ITB_IA	Instruction translation buffer invalidate all	W	105
ITB_IAP	Instruction translation buffer invalidate all process	W	106
ITB_IS	Instruction translation buffer invalidate single	W	107
SIRR	Software interrupt request	R/W	108
ASTRR	Asynchronous system trap request	R/W	109
ASTER	Asynchronous system trap enable	R/W	10A

(continued on next page)

Table 10 (Cont.) Ibox, Mbox, Dcache, and PALtemp Internal Processor Registers

Mnemonic	Register Name	Access	Index (hex)
	Ibox IPRs		
EXC_ADDR	Exception address	R/W	10B
EXC_SUM	Exception summary	R/W0C	10C
EXC_MASK	Exception mask	R	10D
PAL_BASE	Privileged architecture library base address	R/W	10E
ICM	Ibox current mode	R/W	10F
IPLR	Interrupt priority level	R/W	110
INTID	Interrupt ID	R	111
IFAULT_VA_FORM	Formatted faulting virtual address	R	112
IVPTBR	Virtual page table base	R/W	113
HWINT_CLR	Hardware interrupt clear	W	115
SL_XMIT	Serial line transmit	W	116
SL_RCV	Serial line receive	R	117
ICSR	Ibox control and status	R/W	118
IC_FLUSH_CTL	Icache flush control	W	119
ICPERR_STAT	Icache parity error status	R/W1C	11A
PMCTR	Performance counter	R/W	11C

(continued on next page)

Table 10 (Cont.) Ibox, Mbox, Dcache, and PALtemp Internal Processor Registers

Mnemonic	Register Name	Access	Index (hex)
	PALtemp IPRs		
PALtemp0	—	R/W	140
PALtemp1	—	R/W	141
PALtemp2	—	R/W	142
PALtemp3	—	R/W	143
PALtemp4	—	R/W	144
PALtemp5	—	R/W	145
PALtemp6	—	R/W	146
PALtemp7	—	R/W	147
PALtemp8	—	R/W	148
PALtemp9	—	R/W	149
PALtemp10	—	R/W	14A
PALtemp11	—	R/W	14B
PALtemp12	—	R/W	14C
PALtemp13	—	R/W	14D
PALtemp14	—	R/W	14E
PALtemp15	—	R/W	14F
PALtemp16	—	R/W	150
PALtemp17	—	R/W	151
PALtemp18	—	R/W	152
PALtemp19	—	R/W	153
PALtemp20	—	R/W	154
PALtemp21	—	R/W	155
PALtemp22	—	R/W	156
PALtemp23	—	R/W	157

(continued on next page)

Table 10 (Cont.) Ibox, Mbox, Dcache, and PALtemp Internal Processor Registers

Mnemonic	Register Name	Access	Index (hex)
	Mbox IPRs		
DTB_ASN	Dstream translation buffer address space number	W	200
DTB_CM	Dstream translation buffer current mode	W	201
DTB_TAG	Dstream translation buffer tag	W	202
DTB_PTE	Dstream translation buffer page table entry	R/W	203
DTB_PTE_TEMP	Dstream translation buffer page table entry temporary	R	204
MM_STAT	Dstream memory-management fault status	R	205
VA	Faulting virtual address	R	206
VA_FORM	Formatted virtual address	R	207
MVPTBR	Mbox virtual page table base	W	208
DTB_IAP	Dstream translation buffer invalidate all process	W	209
DTB_IA	Dstream translation buffer invalidate all	W	20A
DTB_IS	Dstream translation buffer invalidate single	W	20B
ALT_MODE	Alternate mode	W	20C
CC	Cycle counter	W	20D
CC_CTL	Cycle counter control	W	20E
MCSR	Mbox control	R/W	20F
DC_FLUSH	Dcache flush	W	210
DC_PERR_STAT	Dcache parity error status	R/W1C	212
DC_TEST_CTL	Dcache test tag control	R/W	213
DC_TEST_TAG	Dcache test tag	R/W	214
DC_TEST_TAG_TEMP	Dcache test tag temporary	R/W	215

(continued on next page)

Table 10 (Cont.) Ibox, Mbox, Dcache, and PALtemp Internal Processor Registers

Mnemonic	Register Name	Access	Index (hex)
	Mbox IPRs		
DC_MODE	Dcache mode	R/W	216
MAF_MODE	Miss address file mode	R/W	217

8.2 External Interface Control (Cbox) IPRs

Table 11 summarizes IPRs for controlling Scache, Bcache, system configuration, and logging error information. These IPRs cannot be read or written from the system. They are placed in the 1-MB region of 21164-specific I/O address space ranging from FF FFF0 0000 to FF FFFF FFFF. Any read or write operation to an undefined IPR in this address space produces UNDEFINED behavior. The operating system should not map any address in this region as writable in any mode.

Table 11 Cbox Internal Processor Register Descriptions

Mnemonic	Description	Type[1]	Address
SC_CTL	Scache control	RW	FF FFF0 00A8
SC_STAT	Scache status	R	FF FFF0 00E8
SC_ADDR	Scache address	R	FF FFF0 0188
BC_CONTROL	Bcache control	W	FF FFF0 0128
BC_CONFIG	Bcache configuration	W	FF FFF0 01C8
BC_TAG_ADDR	Bcache tag address	R	FF FFF0 0108
EI_STAT	External interface status	R	FF FFF0 0168
EI_ADDR	External interface address	R	FF FFF0 0148
FILL_SYN	Fill syndrome	R	FF FFF0 0068

[1]BC_CONTROL<01> must be 0 when reading any IPR in this table.

8.3 PALcode Storage Registers

The 21164 Ebox register file has eight extra registers that are called the PALshadow registers. The PALshadow registers overlay R8 through R14 and R25 when the CPU is in PALmode and ICSR<SDE> is set. Thus, PALcode can consider R8 through R14 and R25 as local scratch. PALshadow registers can not be written in the last two cycles of a PALcode flow. The normal state of the CPU is ICSR<SDE> = ON. PALcode disables SDE for the unaligned trap and for error flows.

9 PALcode

Privileged architecture library code (PALcode) is macrocode that provides an architecturally defined operating-system-specific programming interface that is common across all Alpha microprocessors. The actual implementation of PALcode differs for each operating system.

PALcode runs with privileges enabled, instruction stream (Istream) mapping disabled, and interrupts disabled. PALcode has privilege to use five special opcodes that allow functions such as physical data stream (Dstream) references and internal processor register (IPR) manipulation.

PALcode can be invoked by the following events:

- Reset
- System hardware exceptions (MCHK, ARITH)
- Memory-management exceptions
- Interrupts
- CALL_PAL instructions

9.1 PALcode Entry Points

PALcode is invoked at specific entry points. The 21164 has two types of PALcode entry points:

- CALL_PAL entry points are used whenever the Ibox encounters a CALL_PAL instruction in the Istream.
  – Privileged CALL_PAL instructions start at offset 2000.
  – Unprivileged CALL_PAL instructions start at offset 3000.
- Chip-specific trap entry points start PALcode.

9.1.1 PALcode Trap Entry Points

Table 12 shows the PALcode trap entry points and their offset from the PAL_BASE IPR. Entry points are listed from highest to lowest priority.

Table 12 PALcode Trap Entry Points

Entry Name	Offset_{16}	Description
RESET	0000	Reset
IACCVIO	0080	Istream access violation or sign check error on PC
INTERRUPT	0100	Interrupt: hardware, software, and AST
ITBMISS	0180	Istream TBMISS
DTBMISS_SINGLE	0200	Dstream TBMISS
DTBMISS_DOUBLE	0280	Dstream TBMISS during virtual page table entry (PTE) fetch
UNALIGN	0300	Dstream unaligned reference
DFAULT	0380	Dstream fault or sign check error on virtual address
MCHK	0400	Uncorrected hardware error
OPCDEC	0480	Illegal opcode
ARITH	0500	Arithmetic exception
FEN	0580	Floating-point operation attempted with: • Floating-point instructions (LD, ST, and operates) disabled through FPE bit in the ICSR IPR • Floating-point IEEE operation with data type other than S, T, or Q

9.2 Required PALcode Function Codes

Table 13 lists opcodes required for all Alpha implementations. The notation used is *oo.ffff*, where *oo* is the hexadecimal 6-bit opcode and *ffff* is the hexadecimal 26-bit function code.

Table 13 Required PALcode Function Codes

Mnemonic	Type	Function Code
DRAINA	Privileged	00.0002
HALT	Privileged	00.0000
IMB	Unprivileged	00.0086

9.3 Opcodes Reserved for PALcode

Table 14 lists the opcodes reserved by the Alpha architecture for implementation-specific use. These opcodes are privileged and are only available in PALmode. Section 10.0.2 shows the opcodes reserved for PALcode.

Table 14 Opcodes Reserved for PALcode

Opcode	Architecture Mnemonic
1B	PAL1B
1F	PAL1F
1E	PAL1E
19	PAL19
1D	PAL1D

10 Alpha Instruction Summary

This section contains a summary of all Alpha architecture instructions. All values are in hexadecimal radix. Table 15 describes the contents of the Format and Opcode columns that are in Table 16.

Table 15 Instruction Format and Opcode Notation

Instruction Format	Format Symbol	Opcode Notation	Meaning
Branch	Bra	*oo*	*oo* is the 6-bit opcode field.
Floating-point	F-P	*oo.fff*	*oo* is the 6-bit opcode field. *fff* is the 11-bit function code field.
Memory	Mem	*oo*	*oo* is the 6-bit opcode field.
Memory/ function code	Mfc	*oo.ffff*	*oo* is the 6-bit opcode field. *ffff* is the 16-bit function code in the displacement field.
Memory/ branch	Mbr	*oo.h*	*oo* is the 6-bit opcode field. *h* is the high-order 2 bits of the displacement field.
Operate	Opr	*oo.ff*	*oo* is the 6-bit opcode field. *ff* is the 7-bit function code field.
PALcode	Pcd	*oo*	*oo* is the 6-bit opcode field; the particular PALcode instruction is specified in the 26-bit function code field.

Qualifiers for operate instructions are shown in Table 16. Qualifiers for IEEE and VAX floating-point instructions are shown in Tables 19 and 20, respectively.

Table 16 Architecture Instructions

Mnemonic	Format	Opcode	Description
ADDF	F-P	15.080	Add F_floating
ADDG	F-P	15.0A0	Add G_floating
ADDL	Opr	10.00	Add longword
ADDL/V	Opr	10.40	Add longword
ADDQ	Opr	10.20	Add quadword
ADDQ/V	Opr	10.60	Add quadword
ADDS	F-P	16.080	Add S_floating
ADDT	F-P	16.0A0	Add T_floating
AND	Opr	11.00	Logical product
BEQ	Bra	39	Branch if $=$ zero
BGE	Bra	3E	Branch if $\geq$ zero
BGT	Bra	3F	Branch if $>$ zero
BIC	Opr	11.0	Bit clear
BIS	Opr	11.20	Logical sum
BLBC	Bra	38	Branch if low bit clear
BLBS	Bra	3C	Branch if low bit set
BLE	Bra	3B	Branch if $\leq$ zero
BLT	Bra	3A	Branch if $<$ zero
BNE	Bra	3D	Branch if $\neq$ zero
BR	Bra	30	Unconditional branch
BSR	Mbr	34	Branch to subroutine
CALL_PAL	Pcd	00	Trap to PALcode
CMOVEQ	Opr	11.24	CMOVE if $=$ zero
CMOVGE	Opr	11.46	CMOVE if $\geq$ zero
CMOVGT	Opr	11.66	CMOVE if $>$ zero
CMOVLBC	Opr	11.16	CMOVE if low bit clear
CMOVLBS	Opr	11.14	CMOVE if low bit set
CMOVLE	Opr	11.64	CMOVE if $\leq$ zero
CMOVLT	Opr	11.44	CMOVE if $<$ zero
CMOVNE	Opr	11.26	CMOVE if $\neq$ zero
CMPBGE	Opr	10.0F	Compare byte
CMPEQ	Opr	10.2D	Compare signed quadword equal
CMPGEQ	F-P	15.0A5	Compare G_floating equal
CMPGLE	F-P	15.0A7	Compare G_floating less than or equal

(continued on next page)

Table 16 (Cont.) Architecture Instructions

Mnemonic	Format	Opcode	Description
CMPGLT	F-P	15.0A6	Compare G_floating less than
CMPLE	Opr	10.6D	Compare signed quadword less than or equal
CMPLT	Opr	10.4D	Compare signed quadword less than
CMPTEQ	F-P	16.0A5	Compare T_floating equal
CMPTLE	F-P	16.0A7	Compare T_floating less than or equal
CMPTLT	F-P	16.0A6	Compare T_floating less than
CMPTUN	F-P	16.0A4	Compare T_floating unordered
CMPULE	Opr	10.3D	Compare unsigned quadword less than or equal
CMPULT	Opr	10.1D	Compare unsigned quadword less than
CPYS	F-P	17.020	Copy sign
CPYSE	F-P	17.022	Copy sign and exponent
CPYSN	F-P	17.021	Copy sign negate
CVTDG	F-P	15.09E	Convert D_floating to G_floating
CVTGD	F-P	15.0AD	Convert G_floating to D_floating
CVTGF	F-P	15.0AC	Convert G_floating to F_floating
CVTGQ	F-P	15.0AF	Convert G_floating to quadword
CVTLQ	F-P	17.010	Convert longword to quadword
CVTQF	F-P	15.0BC	Convert quadword to F_floating
CVTQG	F-P	15.0BE	Convert quadword to G_floating
CVTQL	F-P	17.030	Convert quadword to longword
CVTQL/SV	F-P	17.530	Convert quadword to longword
CVTQL/V	F-P	17.130	Convert quadword to longword
CVTQS	F-P	16.0BC	Convert quadword to S_floating
CVTQT	F-P	16.0BE	Convert quadword to T_floating
CVTST	F-P	16.2AC	Convert S_floating to T_floating
CVTTQ	F-P	16.0AF	Convert T_floating to quadword
CVTTS	F-P	16.0AC	Convert T_floating to S_floating
DIVF	F-P	15.083	Divide F_floating
DIVG	F-P	15.0A3	Divide G_floating
DIVS	F-P	16.083	Divide S_floating
DIVT	F-P	16.0A3	Divide T_floating
EQV	Opr	11.48	Logical equivalence
EXCB	Mfc	18.0400	Exception barrier
EXTBL	Opr	12.06	Extract byte low
EXTLH	Opr	12.6A	Extract longword high

(continued on next page)

Table 16 (Cont.) Architecture Instructions

Mnemonic	Format	Opcode	Description
EXTLL	Opr	12.26	Extract longword low
EXTQH	Opr	12.7A	Extract quadword high
EXTQL	Opr	12.36	Extract quadword low
EXTWH	Opr	12.5A	Extract word high
EXTWL	Opr	12.16	Extract word low
FBEQ	Bra	31	Floating branch if = zero
FBGE	Bra	36	Floating branch if $\geq$ zero
FBGT	Bra	37	Floating branch if > zero
FBLE	Bra	33	Floating branch if $\leq$ zero
FBLT	Bra	32	Floating branch if < zero
FBNE	Bra	35	Floating branch if $\neq$ zero
FCMOVEQ	F-P	17.02A	FCMOVE if = zero
FCMOVGE	F-P	17.02D	FCMOVE if $\geq$ zero
FCMOVGT	F-P	17.02F	FCMOVE if > zero
FCMOVLE	F-P	17.02E	FCMOVE if $\leq$ zero
FCMOVLT	F-P	17.02C	FCMOVE if < zero
FCMOVNE	F-P	17.02B	FCMOVE if $\neq$ zero
FETCH	Mfc	18.80	Prefetch data
FETCH_M	Mfc	18.A0	Prefetch data, modify intent
INSBL	Opr	12.0B	Insert byte low
INSLH	Opr	12.67	Insert longword high
INSLL	Opr	12.2B	Insert longword low
INSQH	Opr	12.77	Insert quadword high
INSQL	Opr	12.3B	Insert quadword low
INSWH	Opr	12.57	Insert word high
INSWL	Opr	12.1B	Insert word low
JMP	Mbr	1A.0	Jump
JSR	Mbr	1A.1	Jump to subroutine
JSR_COROUTINE	Mbr	1A.3	Jump to subroutine return
LDA	Mem	08	Load address
LDAH	Mem	09	Load address high
LDF	Mem	20	Load F_floating
LDG	Mem	21	Load G_floating
LDL	Mem	28	Load sign-extended longword
LDL_L	Mem	2A	Load sign-extended longword locked
LDQ	Mem	29	Load quadword
LDQ_L	Mem	2B	Load quadword locked
LDQ_U	Mem	0B	Load unaligned quadword

(continued on next page)

Table 16 (Cont.) Architecture Instructions

Mnemonic	Format	Opcode	Description
LDS	Mem	22	Load S_floating
LDT	Mem	23	Load T_floating
MB	Mfc	18.4000	Memory barrier
MF_FPCR	F-P	17.025	Move from floating-point control register
MSKBL	Opr	12.02	Mask byte low
MSKLH	Opr	12.62	Mask longword high
MSKLL	Opr	12.22	Mask longword low
MSKQH	Opr	12.72	Mask quadword high
MSKQL	Opr	12.32	Mask quadword low
MSKWH	Opr	12.52	Mask word high
MSKWL	Opr	12.12	Mask word low
MT_FPCR	F-P	17.024	Move to floating-point control register
MULF	F-P	15.082	Multiply F_floating
MULG	F-P	15.0A2	Multiply G_floating
MULL	Opr	13.00	Multiply longword
MULL/V	Opr	13.40	Multiply longword
MULQ	Opr	13.20	Multiply quadword
MULQ/V	Opr	13.60	Multiply quadword
MULS	F-P	16.082	Multiply S_floating
MULT	F-P	16.0A2	Multiply T_floating
ORNOT	Opr	11.28	Logical sum with complement
RC	Mfc	18.E0	Read and clear
RET	Mbr	1A.2	Return from subroutine
RPCC	Mfc	18.C0	Read process cycle counter
RS	Mfc	18.F000	Read and set
S4ADDL	Opr	10.02	Scaled add longword by 4
S4ADDQ	Opr	10.22	Scaled add quadword by 4
S4SUBL	Opr	10.0B	Scaled subtract longword by 4
S4SUBQ	Opr	10.2B	Scaled subtract quadword by 4
S8ADDL	Opr	10.12	Scaled add longword by 8
S8ADDQ	Opr	10.32	Scaled add quadword by 8
S8SUBL	Opr	10.1B	Scaled subtract longword by 8
S8SUBQ	Opr	10.3B	Scaled subtract quadword by 8
SLL	Opr	12.39	Shift left logical
SRA	Opr	12.3C	Shift right arithmetic
SRL	Opr	12.34	Shift right logical
STF	Mem	24	Store F_floating

(continued on next page)

Table 16 (Cont.) Architecture Instructions

Mnemonic	Format	Opcode	Description
STG	Mem	25	Store G_floating
STS	Mem	26	Store S_floating
STL	Mem	2C	Store longword
STL_C	Mem	2E	Store longword conditional
STQ	Mem	2D	Store quadword
STQ_C	Mem	2F	Store quadword conditional
STQ_U	Mem	0F	Store unaligned quadword
STT	Mem	27	Store T_floating
SUBF	F-P	15.081	Subtract F_floating
SUBG	F-P	15.0A1	Subtract G_floating
SUBL	Opr	10.09	Subtract longword
SUBL/V		10.49	
SUBQ	Opr	10.29	Subtract quadword
SUBQ/V		10.69	
SUBS	F-P	16.081	Subtract S_floating
SUBT	F-P	16.0A1	Subtract T_floating
TRAPB	Mfc	18.00	Trap barrier
UMULH	Opr	13.30	Unsigned multiply quadword high
WMB	Mfc	18.44	Write memory barrier
XOR	Opr	11.40	Logical difference
ZAP	Opr	12.30	Zero bytes
ZAPNOT	Opr	12.31	Zero bytes not

10.0.1 Opcodes Reserved for Digital

Table 17 lists opcodes reserved for Digital.

Table 17 Opcodes Reserved for Digital

Mnemonic	Opcode	Mnemonic	Opcode	Mnemonic	Opcode
OPC01	01	OPC05	05	OPC0B	0B
OPC02	02	OPC06	06	OPC0C	0C
OPC03	03	OPC07	07	OPC0D	0D
OPC04	04	OPC0A	0A	OPC14	14

10.0.2 Opcodes Reserved for PALcode

Table 18 lists the 21164-specific instructions. For more information, refer to the *Alpha 21164 Microprocessor Hardware Reference Manual*.

Table 18 Opcodes Reserved for PALcode

21164 Mnemonic	Opcode	Architecture Mnemonic	Function
HW_LD	1B	PAL1B	Performs Dstream load instructions.
HW_ST	1F	PAL1F	Performs Dstream store instructions.
HW_REI	1E	PAL1E	Returns instruction flow to the program counter (PC) pointed to by EXC_ADDR internal processor register (IPR).
HW_MFPR	19	PAL19	Accesses the Ibox, Mbox, and Dcache IPRs.
HW_MTPR	1D	PAL1D	Accesses the Ibox, Mbox, and Dcache IPRs.

10.1 IEEE Floating-Point Instructions

Table 19 lists the hexadecimal value of the 11-bit function code field for the IEEE floating-point instructions, with and without qualifiers. The opcode for these instructions is 16_{16}.

Table 19 IEEE Floating-Point Instruction Function Codes

Mnemonic	None	/C	/M	/D	/U	/UC	/UM	/UD
ADDS	080	000	040	0C0	180	100	140	1C0
ADDT	0A0	020	060	0E0	1A0	120	160	1E0
CMPTEQ	0A5							
CMPTLT	0A6							
CMPTLE	0A7							
CMPTUN	0A4							
CVTQS	0BC	03C	07C	0FC				
CVTQT	0BE	03E	07E	0FE				
CVTTS	0AC	02C	06C	0EC	1AC	12C	16C	1EC

(continued on next page)

Table 19 (Cont.) IEEE Floating-Point Instruction Function Codes

Mnemonic	None	/C	/M	/D	/U	/UC	/UM	/UD
DIVS	083	003	043	0C3	183	103	143	1C3
DIVT	0A3	023	063	0E3	1A3	123	163	1E3
MULS	082	002	042	0C2	182	102	142	1C2
MULT	0A2	022	062	0E2	1A2	122	162	1E2
SUBS	081	001	041	0C1	181	101	141	1C1
SUBT	0A1	021	061	0E1	1A1	121	161	1E1

Mnemonic	/SU	/SUC	/SUM	/SUD	/SUI	/SUIC	/SUIM	/SUID
ADDS	580	500	540	5C0	780	700	740	7C0
ADDT	5A0	520	560	5E0	7A0	720	760	7E0
CMPTEQ	5A5							
CMPTLT	5A6							
CMPTLE	5A7							
CMPTUN	5A4							
CVTQS					7BC	73C	77C	7FC
CVTQT					7BE	73E	77E	7FE
CVTTS	5AC	52C	56C	5EC	7AC	72C	76C	7EC
DIVS	583	503	543	5C3	783	703	743	7C3
DIVT	5A3	523	563	5E3	7A3	723	763	7E3
MULS	582	502	542	5C2	782	702	742	7C2
MULT	5A2	522	562	5E2	7A2	722	762	7E2
SUBS	581	501	541	5C1	781	701	741	7C1
SUBT	5A1	521	561	5E1	7A1	721	761	7E1

Mnemonic	None	/S
CVTST	2AC	6AC

Mnemonic	None	/C	/V	/VC	/SV	/SVC	/SVI	/SVIC
CVTTQ	0AF	02F	1AF	12F	5AF	52F	7AF	72F

Mnemonic	D	/VD	/SVD	/SVID	/M	/VM	/SVM	/SVIM
CVTTQ	0EF	1EF	5EF	7EF	06F	16F	56F	76F

Programming Note

Because underflow cannot occur for CMPT*xx*, there is no difference in function or performance between CMPT*xx*/S and CMPT*xx*/SU. It is intended that software generate CMPT*xx*/SU in place of CMPT*xx*/S.

In the same manner, CVTQS and CVTQT can take an inexact result trap, but not an underflow. Because there is no encoding for a CVTQ*x*/SI instruction, it is intended that software generate CVTQ*x*/SUI in place of CVTQ*x*/SI.

10.2 VAX Floating-Point Instructions

Table 20 lists the hexadecimal value of the 11-bit function code field for the VAX floating-point instructions. The opcode for these instructions is 15_{16}.

Table 20 VAX Floating-Point Instruction Function Codes

Mnemonic	None	/C	/U	/UC	/S	/SC	/SU	/SUC
ADDF	080	000	180	100	480	400	580	500
CVTDG	09E	01E	19E	11E	49E	41E	59E	51E
ADDG	0A0	020	1A0	120	4A0	420	5A0	520
CMPGEQ	0A5				4A5			
CMPGLT	0A6				4A6			
CMPGLE	0A7				4A7			
CVTGF	0AC	02C	1AC	12C	4AC	42C	5AC	52C
CVTGD	0AD	02D	1AD	12D	4AD	42D	5AD	52D
CVTQF	0BC	03C						
CVTQG	0BE	03E						
DIVF	083	003	183	103	483	403	583	503
DIVG	0A3	023	1A3	123	4A3	423	5A3	523
MULF	082	002	182	102	482	402	582	502
MULG	0A2	022	1A2	122	4A2	422	5A2	522
SUBF	081	001	181	101	481	401	581	501
SUBG	0A1	021	1A1	121	4A1	421	5A1	521

Mnemonic	None	/C	/V	/VC	/S	/SC	/SV	/SVC
CVTGQ	0AF	02F	1AF	12F	4AF	42F	5AF	52F

10.3 Opcode Summary

Table 21 lists all Alpha opcodes from 00 (CALL_PAL) through 3F (BGT). In the table, the column headings that appear over the instructions have a granularity of 8_{16}. The rows beneath the Offset column supply the individual hexadecimal number to resolve that granularity.

If an instruction column has a 0 in the right (low) hexadecimal digit, replace that 0 with the number to the left of the backslash (\) in the Offset column on the instruction's row. If an instruction column has an 8 in the right (low) hexadecimal digit, replace that 8 with the number to the right of the backslash in the Offset column.

For example, the third row (2/A) under the 10_{16} column contains the symbol INTS*, representing the all-integer shift instructions. The opcode for those instructions would then be 12_{16} because the 0 in 10 is replaced by the 2 in the Offset column. Likewise, the third row under the 18_{16} column contains the symbol JSR*, representing all jump instructions. The opcode for those instructions is 1A because the 8 in the heading is replaced by the number to the right of the backslash in the Offset column.

The instruction format is listed under the instruction symbol.

Table 21 Opcode Summary

Offset	00	08	10	18	20	28	30	38
0/8	PAL* (pal)	LDA (mem)	INTA* (op)	MISC* (mem)	LDF (mem)	LDL (mem)	BR (br)	BLBC (br)
1/9	Res	LDAH (mem)	INTL* (op)	\PAL\	LDG (mem)	LDQ (mem)	FBEQ (br)	BEQ (br)
2/A	Res	Res	INTS* (op)	JSR* (mem)	LDS (mem)	LDL_L (mem)	FBLT (br)	BLT (br)
3/B	Res	LDQ_U (mem)	INTM* (op)	\PAL\	LDT (mem)	LDQ_L (mem)	FBLE (br)	BLE (br)
4/C	Res	Res	Res	Res	STF (mem)	STL (mem)	BSR (br)	BLBS (br)
5/D	Res	Res	FLTV* (op)	\PAL\	STG (mem)	STQ (mem)	FBNE (br)	BNE (br)
6/E	Res	Res	FLTI* (op)	\PAL\	STS (mem)	STL_C (mem)	FBGE (br)	BGE (br)
7/F	Res	STQ_U (mem)	FLTL* (op)	\PAL\	STT (mem)	STQ_C (mem)	FBGT (br)	BGT (br)

Symbol	**Meaning**
FLTI*	IEEE floating-point instruction opcodes
FLTL*	Floating-point operate instruction opcodes
FLTV*	VAX floating-point instruction opcodes
INTA*	Integer arithmetic instruction opcodes
INTL*	Integer logical instruction opcodes
INTM*	Integer multiply instruction opcodes
INTS*	Integer shift instruction opcodes
JSR*	Jump instruction opcodes
MISC*	Miscellaneous instruction opcodes
PAL*	PALcode instruction (CALL_PAL) opcodes
\PAL\	Reserved for PALcode
Res	Reserved for Digital

10.4 Required PALcode Function Codes

Table 22 lists opcodes required for all Alpha implementations. The notation used is *oo.ffff*, where *oo* is the hexadecimal 6-bit opcode and *ffff* is the hexadecimal 26-bit function code.

Table 22 Required PALcode Function Codes

Mnemonic	Type	Function Code
DRAINA	Privileged	00.0002
HALT	Privileged	00.0000
IMB	Unprivileged	00.0086

11 Electrical Data

This chapter describes the electrical characteristics of the 21164 component and its interface pins. It is organized as follows:

- Electrical characteristics
- dc characteristics
- Clocking scheme
- ac characteristics
- Power supply considerations

11.1 Electrical Characteristics

Table 23 lists the maximum ratings for the 21164.

Table 23 Alpha 21164 Absolute Maximum Ratings

Characteristics	Ratings
Storage temperature	–55°C to 125°C (–67°F to 257°F)
Junction temperature	15°C to 90°C (59°F to 194°F)
Supply voltage	**Vss** –0.5 V, **Vdd** 3.6 V
Input or output applied[1]	–0.5 V to 6.3 V
Typical worst case power **@Vdd** = 3.3 V Frequency = 266 MHz Frequency = 300 MHz	 46 W 51 W

[1]Refer to Section 11.5.2.

Caution

Stress beyond the absolute maximum rating can cause permanent damage to the 21164. Exposure to absolute maximum rating conditions for extended periods of time can affect the 21164 reliability.

11.2 dc Characteristics

The 21164 is designed to run in a CMOS/TTL environment. The 21164 is tested and characterized in a CMOS environment.

11.2.1 Power Supply

The **Vss** pins are connected to 0.0 V, and the **Vdd** pins are connected to 3.3 V, ±5%.

11.2.2 Input Signal Pins

Nearly all input signals are ordinary CMOS inputs with standard TTL levels (see Table 24). (See Section 11.3.1 for a description of an exception—**osc_clk_in_h,l**.)

After power has been applied, input and bidirectional pins can be driven to a maximum dc voltage of 6.3 V (6.8 V for 1 ns) without harming the 21164. (It is not necessary to use static RAMs with 3.3-V outputs.)

11.2.3 Output Signal Pins

Output pins are ordinary 3.3-V CMOS outputs. Although output signals are rail-to-rail, timing is specified to $\frac{Vdd}{2}$.

Bidirectional pins are either input or output pins, depending on control timing. When functioning as output pins, they are ordinary 3.3-V CMOS outputs.

Table 24 shows the CMOS dc input and output pins.

Table 24 CMOS dc Input/Output Characteristics

Parameter		Requirements			
Symbol	**Description**	**Min.**	**Max.**	**Units**	**Test Conditions**
Vih	High-level input voltage	2.0	—	V	—
Vil	Low-level input voltage	—	0.8	V	—
Voh	High-level output voltage	2.4	—	V	**Ioh** = –6.0 mA
Vol	Low-level output voltage	—	0.4	V	**Iol** = 6.0 mA
Iil_pd	Input with pull-down leakage current	—	±50	μA	**Vin** = 0 V
Iih_pd	Input with pull-down current	—	200	μA	**Vin** = 2.4 V
Iil_pu	Input with pull-up current	—	–800	μA	**Vin** = 0.4 V
Iih_pu	Input with pull-up leakage current	—	±50	μA	**Vin** = **Vdd** V
Iozl_pd	Output with pull-down leakage current (tristate)	—	±50	μA	**Vin** = 0 V
Iozh_pd	Output with pull-down current (tristate)	—	200	μA	**Vin** = 2.4 V
Iozl_pu	Output with pull-up current (tristate)	—	–800	μA	**Vin** = 0.4 V
Iozh_pu	Output with pull-up leakage current (tristate)	—	±50	μA	**Vin** = **Vdd** V
Idd	Peak power supply current	—	18	A	**Vdd** = 3.465 V Frequency = 266 MHz
Idd	Peak power supply current	—	20	A	**Vdd** = 3.465 V Frequency = 300 MHz

Most pins have low current pull-down devices to **Vss**. However, two pins have a pull-up device to **Vdd**. The pull-downs (or pull-ups) are always enabled. This means that some current will flow from the 21164 (if the pin has a pull-up device) or into the 21164 (if the pin has a pull-down device) even when the pin is in the high-impedance state. All pins have pull-down devices, except for the pins in the following table:

Signal Name	Notes
tms_h	Has a pull-up device
tdi_h	Has a pull-up device
osc_clk_in_h	50 Ω to **Vterm** ($\approx\frac{Vdd}{2}$) (See Figure 13)
osc_clk_in_l	50 Ω to **Vterm** ($\approx\frac{Vdd}{2}$) (See Figure 13)
temp_sense	150 Ω to **Vss**

11.3 Clocking Scheme

The differential input clock signals **osc_clk_in_h,l** run at two times the internal frequency of the time base for the 21164. Input clocks are divided by two onchip to generate a 50% duty cycle clock for internal distribution. The output signal **cpu_clk_out_h** toggles with an unspecified propagation delay relative to the transitions on **osc_clk_in_h,l**.

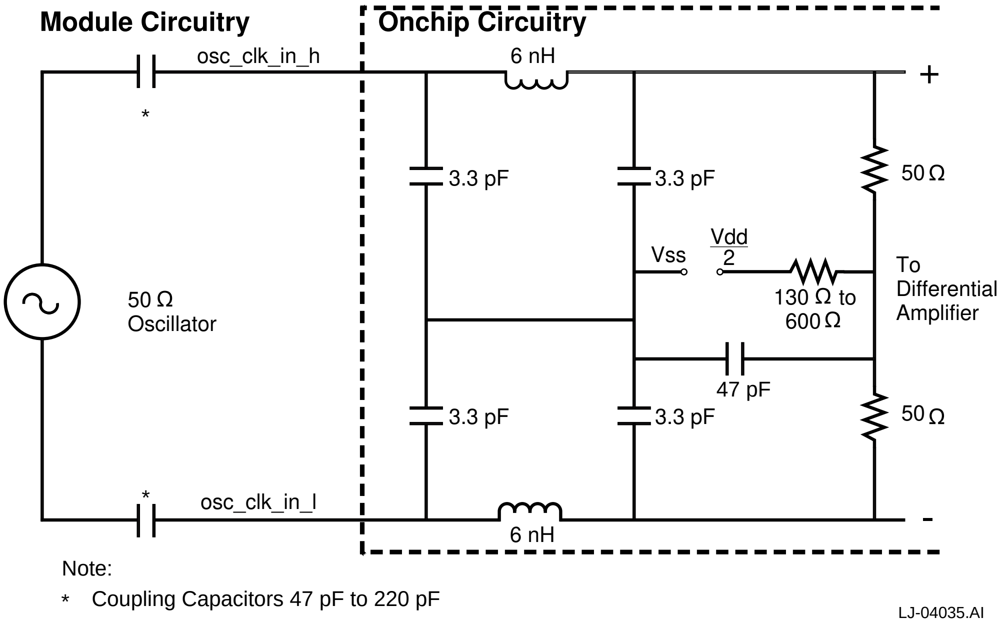
System designers have a choice of two system clocking schemes to run the 21164 synchronous to the system:

1. The 21164 generates and drives out a system clock, **sys_clk_out1_h,l**. It runs synchronous to the internal clock at a selected ratio of the internal clock frequency. There is a small clock skew between the internal clock and **sys_clk_out1_h,l**.
2. The 21164 synchronizes to a system clock, **ref_clk_in_h**, supplied by the system. The **ref_clk_in_h** clock runs at a selected ratio of the 21164 internal clock frequency. The internal clock is synchronized to the reference clock by an onchip digital phase-locked loop (DPLL).

11.3.1 Input Clocks

The differential input clocks **osc_clk_in_h,l** provide the time base for the chip when **dc_ok_h** is asserted. These pins are self-biasing, and must be capacitively coupled to the clock source on the module, or they can be directly driven. The terminations on these signals are designed to be compatible with system oscillators of arbitrary dc bias. The oscillator must have a duty cycle of 60%/40% or tighter. Figure 13 shows the input network and the schematic equivalent of **osc_clk_in_h,l** terminations.

Figure 13 osc_clk_in_h,l Input Network and Terminations



Ring Oscillator

When signal **dc_ok_h** is deasserted, the clock outputs follow the internal ring oscillator. The 21164 runs off the ring oscillator, just as it would when an external clock is applied. The frequency of the ring oscillator varies from chip to chip within a range of 10 MHz to 100 MHz. This corresponds to an internal CPU clock frequency range of 5 MHz to 50 MHz. The system clock divisor is forced to 8, and the **sys_clk_out2** delay is forced to 3.

Clock Sniffer

A special onchip circuit monitors the **osc_clk_in** pins and detects when input clocks are not present. When activated, this circuit switches the 21164 clock generator from the **osc_clk_in** pins to the internal ring oscillator. This happens independently of the state of the **dc_ok_h** pin. The **dc_ok_h** pin functions normally if clocks are present on the **osc_clk_in** pins.

11.3.2 Clock Termination and Impedance Levels

In Figure 13, the clock is designed to approximate a 50- Ω termination for the purpose of impedance matching for those systems that drive input clocks across long traces. The clock input pins appear as a 50-Ω series termination resistor connected to a high impedance voltage source. The voltage source produces a nominal voltage value of $\frac{Vdd}{2}$. The source has an impedance of between 130 Ω and 600 Ω. This voltage is called the self-bias voltage and sources current when the applied voltage at the clock input pins is less than the self-bias voltage. It sinks current when the applied voltage exceeds the self-bias voltage. This high impedance bias driver allows a clock source of arbitrary dc bias to be ac coupled to the 21164. The peak-to-peak amplitude of the clock source must be between 0.6 V and 3.0 V. Either a square-wave or a sinusoidal source may be used. Full-rail clocks may be driven by testers. In any case, the oscillator should be ac coupled to the **osc_clk_in_h,l** inputs by 47 pF through 220 pF capacitors.

11.3.3 ac Coupling

Using series coupling (blocking) capacitors renders the 21164 clock input pins insensitive to the oscillator's dc level. When connected this way, oscillators with any dc offset relative to **Vss** can be used provided they can drive a signal into the **osc_clk_in_h,l** pins with a peak-to-peak level of at least 600 mV, but no greater than 3.0 V peak to peak.

The value of the coupling capacitor is not overly critical. However, it should be sufficiently low impedance at the clock frequency so that the oscillator's output signal (when measured at the **osc_clk_in_h,l** pins) is not attenuated below the 600 mV peak-to-peak lower limit. For sine waves or oscillators producing nearly sinusoidal (pseudo square wave) outputs, 220 pF is recommended at 533.3 MHz (266.6 MHz $\times$ 2). A high quality dielectric such as NPO is required to avoid dielectric losses.

Table 25 shows the input clock specification.

Table 25 Input Clock Specification

Signal Parameter	Nominal Bin[1]	Unit
osc_clk_in_h,l symmetry	50 ± 10	%
osc_clk_in_h,l minimum voltage	0.6	V (peak-to-peak)
osc_clk_in_h,l Z input	50	Ω

[1]Minimum clock frequency = 50.0 MHz
Maximum clock frequency = Tcycle (Tcycle = 266.6 MHz or 300 MHz)

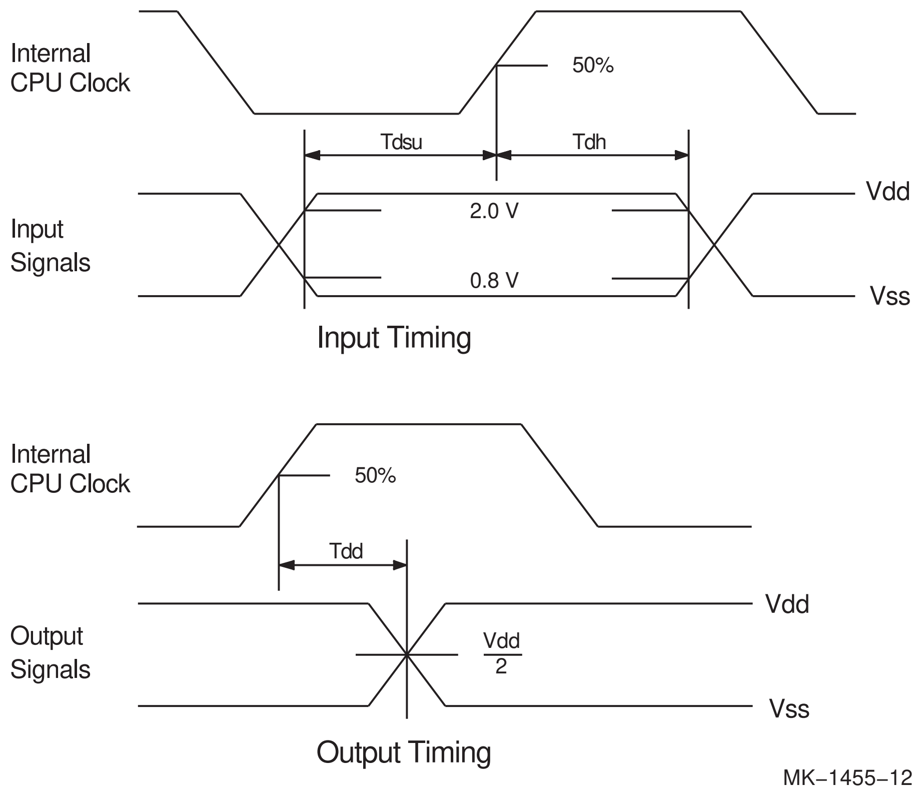
11.4 ac Characteristics

This section describes the ac timing specifications for the 21164.

11.4.1 Test Configuration

All input timing is specified relative to the crossing of standard TTL input levels of 0.8 V and 2.0 V. Output timing is to the nominal CMOS switch point of $\frac{Vdd}{2}$ (see Figure 14).

Figure 14 Input/Output Pin Timing



Because the speed and complexity of microprocessors has increased substantially over the years, it is necessary to change the way they are tested. Traditional assumptions that all loads can be lumped into some accumulation of capacitance cannot be employed any more. Rather, the model of a transmission line with discrete loads is a much more realistic approach for current test technology.

Typically, printed circuit board (PCB) etch has a characteristic impedance of approximately 75 Ω. This may vary from 60 Ω to 90 Ω with tolerances. If the line is driven in the electrical center, the load could be as low as 30 Ω. Therefore, a characteristic impedance range of 30 Ω to 90 Ω could be experienced.

The 21164 output drivers are designed with typical printed circuit board applications in mind rather than trying to accommodate a 40-pF test load specification. As such, it ''launches'' a voltage step into a characteristic impedance, ranging from 30 Ω to 90 Ω.

To prevent signal quality problems due to overshoot or ringing, ''near end'' terminated transmission line design rules are used. By combining the source impedance of the driver transistors with an additional 20-Ω onchip resistor, a source impedance of approximately 40 Ω is achieved. Additionally, a load value of 10 pF, when added to the PCB etch delays, provides a realistic estimate of actual system timing. When employing this test configuration, the signal at the end of the line will transition cleanly through the TTL input specification range of 0.8 V to 2.0 V without plateaus, or reversal into the range.

11.4.2 Pin Timing

The following sections describe Bcache loop timing, sys_clk-based system timing, and reference clock-based system timing.

Backup Cache Loop Timing

The 21164 can be configured to support an optional offchip backup cache (Bcache). Private Bcache read or write (Scache victims) transactions initiated by the 21164 are independent of the system clocking scheme. Bcache loop timing must be an integer multiple of the 21164 cycle time.

Table 26 lists the Bcache loop timing.

Table 26 Bcache Loop Timing

Signal	Specification	Value	Name
data_h<127:0>	Input setup	1.1 ns	**Tdsu**
data_h<127:0>	Input hold	0.0 ns	**Tdh**
index_h<25:4>	Output delay	**Tdd** + 0.4 ns[1]	**Tiod**
index_h<25:4>	Output hold time	**Tmdd**	**Tioh**
data_h<127:0>	Output delay	**Tdd** + **Tcycle** + 0.4 ns[1]	**Tdod**
data_h<127:0>	Output hold	**Tmdd** + **Tcycle**	**Tdoh**

[1]The value 0.4 ns accounts for onchip driver and clock skew.

Outgoing Bcache index and data signals are driven off the internal clock edge and the incoming Bcache tag and data signals are latched on the same internal clock edge. Table 27 shows the output driver characteristics.

Table 27 Output Driver Characteristics

Specification	40-pF Load	10-pF Load	Name
Maximum driver delay	2.6 ns	1.6 ns	**Tdd**
Minimum driver delay	1.0 ns	1.0 ns	**Tmdd**

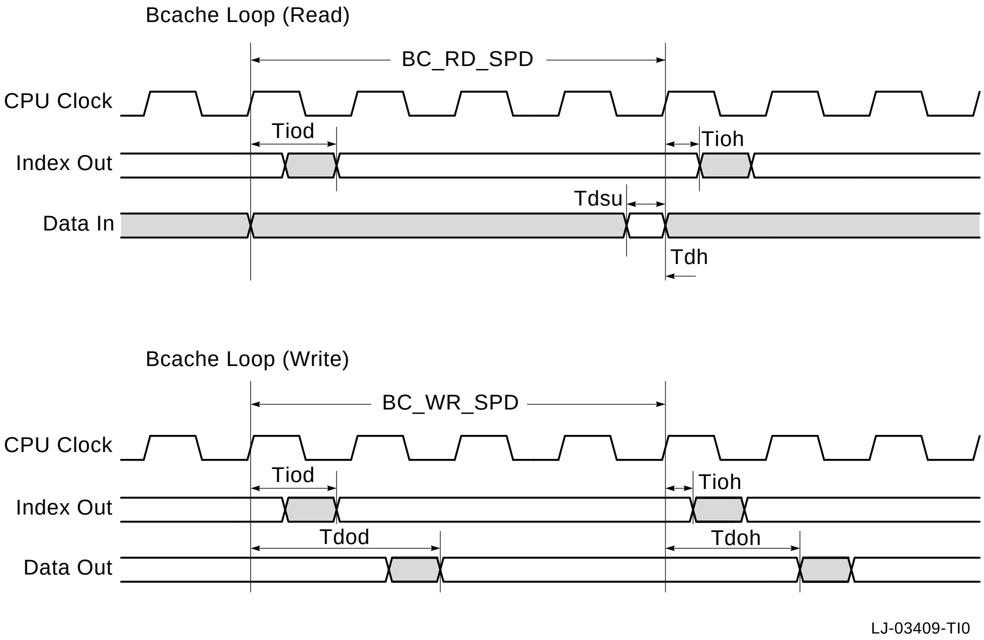
Output pin timing is specified for lumped 40-pF and 10-pF loads. In some cases, the circuit may have loads higher than 40 pF. The 21164 can safely drive higher loads provided the average charging or discharging current from each pin is 10 mA or less. The following equation can be used to determine the maximum capacitance that can be safely driven by each pin:

> Cmax (in pF) = 3t, where t is the waveform period (measured from rising to rising or falling to falling edge), in nanoseconds.

For example, if the waveform appearing on a given I/O pin has a 20.4-ns period, it can safely drive up to and including 61 pF.

Figure 15 shows the Bcache read and write timing.

Figure 15 Bcache Timing



sys_clk-Based Systems

All timing is specified relative to the rising edge of the internal CPU clock.

Table 28 shows 21164 system clock **sys_clk_out1_h,l** output timing. Setup and hold times are specified independent of the relative capacitive loading of **sys_clk_out1_h,l**, **addr_h<39:4>**, **data_h<127:0>**, and **cmd_h<3:0>** signals. The **ref_clk_in_h** signal must be tied to **Vdd** for proper operation.

Table 28 Alpha 21164 System Clock Output Timing (sysclk=$T_{\varnothing}$)

Signal	Specification	Value	Name
sys_clk_out1_h,l	Output delay	**Tdd**	**Tsysd**
sys_clk_out1_h,l	Minimum output delay	**Tmdd**	**Tsysdm**
data_bus_req_h, **data_h<127:0>**, **addr_h<39:4>**	Input setup	1.1 ns	**Tdsu**
data_bus_req_h, **data_h<127:0>**, **addr_h<39:4>**	Input hold	0 ns	**Tdh**
addr_h<39:4>	Output delay	**Tdd** + 0.4 ns[1]	**Taod**
addr_h<39:4>	Output hold time	**Tmdd**	**Taoh**
data_h<127:0>	Output delay	**Tdd** + **Tcycle** + 0.4 ns[1]	**Tdod**[2]
data_h<127:0>	Output hold time	**Tmdd** + **Tcycle**[1]	**Tdoh**[2]
	Non-Pipe_Latch Mode		
addr_bus_req_h	Input setup	3.8 ns	**Tabrsu**
addr_bus_req_h	Input hold	–1.0 ns	**Tabrh**
dack_h	Input setup	3.4 ns	**Tntacksu**
cack_h	Input setup	3.7 ns	**Tntcacksu**
cack, **dack**	Input hold	–1.0 ns	**Tntackh**
	Pipe_Latch Mode[3]		
addr_bus_req_h, **cack_h**, **dack_h**	Input setup	1.1 ns	**Ttacksu**
addr_bus_req_h, **cack_h**, **dack_h**	Input hold	0 ns	**Ttackh**

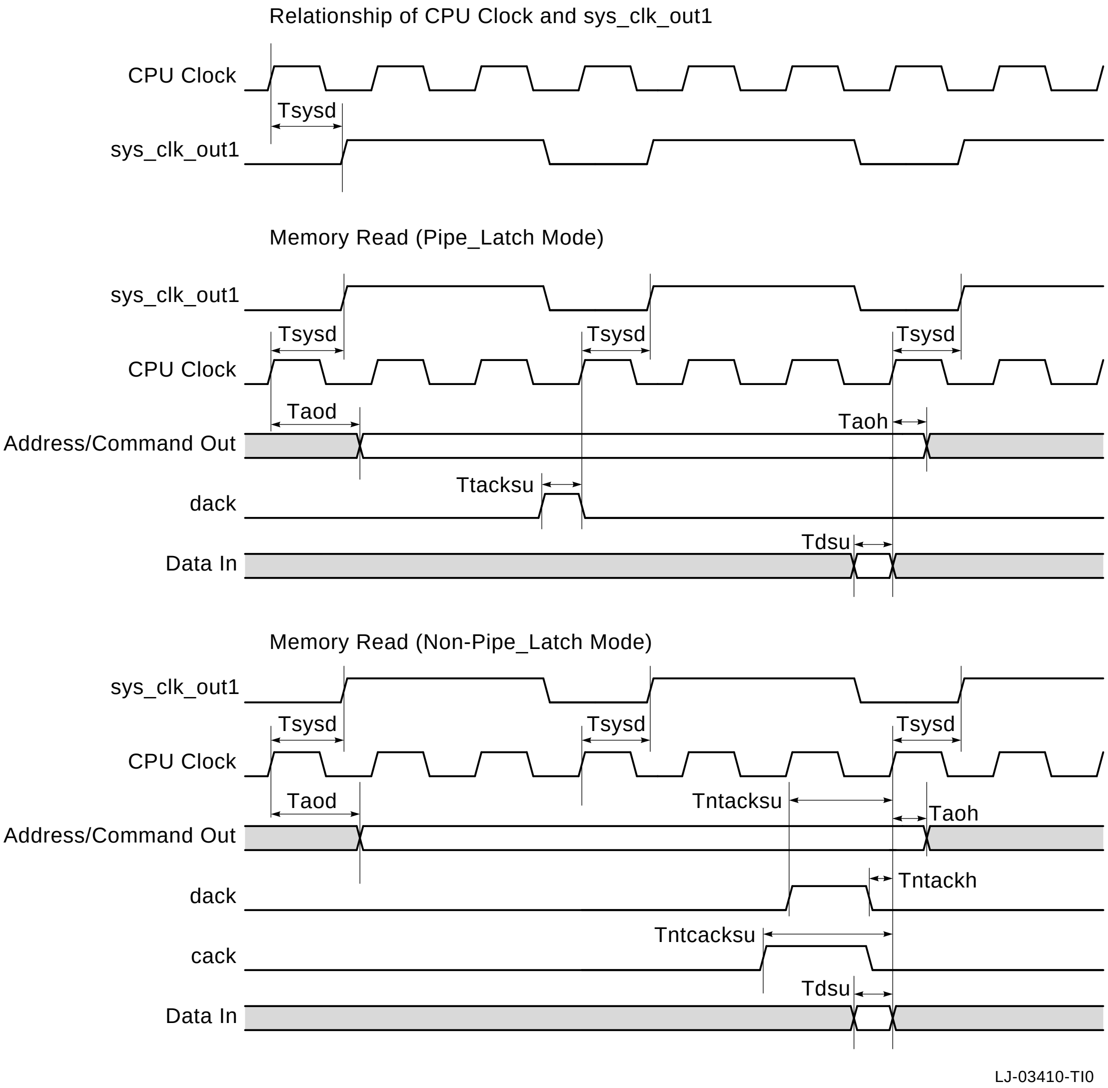
[1]The value 0.4 ns accounts for onchip driver and clock skew.

[2]For all write transactions initiated by the 21164, data is driven one CPU cycle after the **sys_clk_out1** or **index_h<25:4>** pins.

[3]In pipe_latch mode, control signals are piped onchip for one **sys_clk_out1_h,l** before usage.

Figure 16 shows sys_clk system timing.

Figure 16 sys_clk System Timing



Reference Clock-Based Systems

Systems that generate their own system clock expect the 21164 to synchronize its **sys_clk_out1_h,l** outputs to their system clock. The 21164 uses a digital phase-locked loop (DPLL) to synchronize its **sys_clk_out1** signals to the system clock that is applied to the **ref_clk_in_h** signal.

Table 29 shows all timing relative to the rising edge of **ref_clk_in_h**.

Table 29 Alpha 21164 Reference Clock Input Timing

Signal	Specification	Value	Name
data_bus_req_h, **data_h<127:0>**, **addr_h<39:4>**	Input setup	1.1 ns	**Tdsu**
data_bus_req_h, **data_h<127:0>**, **addr_h<39:4>**	Input hold	0.5 x **Tcycle**	**Troh**
addr_h<39:4>	Output delay	**Tdd** + 0.5 x **Tcycle** + 0.9 ns[1]	**Traod**
addr_h<39:4>	Output hold time	**Tmdd**	**Traoh**
data_h<127:0>	Output delay	**Tdd** + 1.5 x **Tcycle** + 0.9 ns[1]	**Trdod**[2]
data_h<127:0>	Output hold time	**Tmdd** + **Tcycle**	**Trdoh**[2]
		Non-Pipe_Latch Mode	
addr_bus_req_h	Input setup	3.8 ns	**Tntrabrsu**
addr_bus_req_h	Input hold	0.5 x **Tcycle**	**Tntrabrh**
dack_h	Input setup	3.3 ns	**Tntracksu**
cack_h	Input setup	3.7 ns	**Tntrcacksu**
cack_h, **dack_h**	Input hold	(0.5 x **Tcycle**)	**Tntrackh**

[1]The value 0.9 ns accounts for onchip skews that include 0.4 ns for driver and clock skew, phase detector skews due to circuit delay (0.2 ns), and delay in **ref_clk_in_h** due to the package (0.3 ns).

[2]For all write transactions initiated by the 21164, data is driven one CPU cycle later.

(continued on next page)

Table 29 (Cont.) Alpha 21164 Reference Clock Input Timing

Signal	Specification	Value	Name
		Pipe_Latch Mode[3]	
addr_bus_req_h, **cack_h**, **dack_h**	Input setup	1.1 ns	**Ttracksu**
addr_bus_req_h, **cack_h**, **dack_h**	Input hold	0.5 x **Tcycle**	**Ttrackh**

[3]In pipe_latch mode, control signals are piped onchip for one **sys_clk_out1_h,l** before usage.

11.4.3 Digital Phase-Locked Loop

Figure 17 and Table 30 describe the digital phase-locked loop (DPLL) stages of operation.

Figure 17 ref_clk System Timing

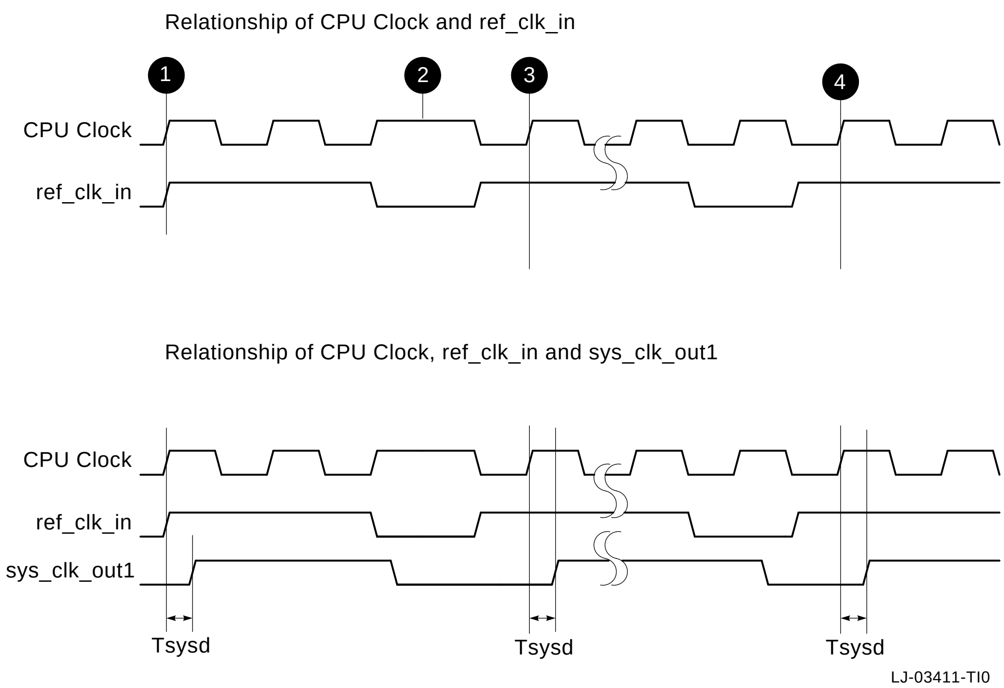


Table 30 ref_clk System Timing Stages

Stage	Description
❶	The internal CPU clock rising edge coincides with the rising edge of **ref_clk_in_h**.
❷	The DPLL causes the internal CPU clock to stretch for one phase (1 cycle of **osc_clk_in_h,l**).
❸	The stretch causes **ref_clk_in_h** to lead the internal CPU clock by one phase.
❹	The CPU clock is always slightly faster than the external **ref_clk_in_h** and gains on **ref_clk_in_h** over time. Eventually the gain equals one phase and a new stretch phase follows.

Although systems that supply a **ref_clk_in_h** do not use **sys_clk_out1_h,l**, a relationship between the two signals exists, just as in the **sys_clk**-based systems, because the 21164 uses **sys_clk_out1_h,l** internally to determine timing during system transactions.

11.4.4 Timing—Additional Signals

This section lists timing for all other signals.

Asynchronous Input Signals

The following is a list of the asynchronous input signals:

clk_mode_h	**dc_ok_h**	**ref_clk_in_h**	
sys_reset_l[1]			
perf_mon_h[2]			
irq_h<3:0>[2]	**mch_hlt_irq_h**[2]	**pwr_fail_irq_h**[2]	**sys_mch_chk_irq_h**[2]

[1] Signal **sys_reset_l** may be deasserted synchronously.

[2] These signals can also be used synchronously.

Miscellaneous Signals

Table 31 and Table 32 list the timing for miscellaneous input-only and output-only signals. All timing is expressed in nanoseconds.

Table 31 Input Timing for sys_clk_out- or ref_clk_in-Based Systems

		Value		Name	
Signal	Specification	sys_clk_out	ref_clk_in	sys_clk_out	ref_clk_in
cfail_h, **fill_h**, **fill_error_h**, **fill_id_h**, **fill_nocheck_h**, **idle_bc_h**, **shared_h**, **system_lock_flag_h** **irq_h<3:0>**, **mch_hlt_irq_h**, **pwr_fail_irq_h**, **sys_mch_chk_irq_h** Testability pins: **port_mode_h**, **srom_data_h**, **srom_present_l**	Input setup	1.1 ns	1.1 ns	**Tdsu**	**Tdsu**
cfail_h, **fill_h**, **fill_error_h**, **fill_id_h**, **fill_nocheck_h**, **idle_bc_h**, **shared_h**, **system_lock_flag_h** **irq_h<3:0>**, **mch_hlt_irq_h**, **pwr_fail_irq_h**, **sys_mch_chk_irq_h** **sys_reset_l** Testability pins: **port_mode_h**, **srom_data_h**, **srom_present_l**	Input hold	0 ns	0.5 ∗**Tcycle**	**Tdh**	**Troh**

Table 32 Output Timing for sys_clk_out- or ref_clk_in-Based Systems

		Clocking System Value		Clocking System Name	
Signal	Specification	sys_clk_out	ref_clk_in	sys_clk_out	ref_clk_in
Unidirectional Signals					
addr_res_h, **int4_valid_h**,[1] **scache_set_h**, **srom_clk_h**, **srom_oe_l**, **victim_pending_h**	Output delay	**Tdd**+0.4 ns	**Tdd**+0.5∗**Tcycle**+0.9 ns	**Taod**	**Traod**

[1]Read transaction

(continued on next page)

Table 32 (Cont.) Output Timing for sys_clk_out- or ref_clk_in-Based Systems

		Clocking System Value		Clocking System Name	
Signal	**Specification**	**sys_clk_out**	**ref_clk_in**	**sys_clk_out**	**ref_clk_in**
Unidirectional Signals					
addr_res_h, **int4_valid_h**,[1] **scache_set_h**, **srom_clk_h**, **srom_oe_l**, **victim_pending_h**	Output hold	**Tmdd**	**Tmdd**	**Taoh**	**Traoh**
int4_valid_h[2]	Output delay	**Tdd**+**Tcycle**+0.4 ns	**Tdd**+1.5∗**Tcycle**+0.9 ns	**Tdod**	**Trdod**
int4_valid_h[2]	Output hold	**Tmdd+Tcycle**	**Tmdd+Tcycle**	**Tdoh**	**Trdoh**
Bidirectional Signals					
Input mode:					
addr_cmd_par_h, **cmd_h**, **data_check_h**,[1] **tag_ctl_par_h**,[3] **tag_dirty_h**,[3] **tag_shared_h**[3]	Input setup	1.1 ns	1.1 ns	**Tdsu**	**Tdsu**
addr_cmd_par_h, **cmd_h**, **data_check_h**,[1] **tag_ctl_par_h**,[3] **tag_dirty_h**,[3] **tag_shared_h**[3]	Input hold	0 ns	0.5∗**Tcycle**	**Tdh**	**Tsdadh**

[1]Read transaction
[2]Write transaction
[3]Fills from memory

(continued on next page)

Table 32 (Cont.) Output Timing for sys_clk_out- or ref_clk_in-Based Systems

		Clocking System Value		Clocking System Name	
Signal	**Specification**	**sys_clk_out**	**ref_clk_in**	**sys_clk_out**	**ref_clk_in**
Bidirectional Signals					
Output mode:					
addr_cmd_par_h, **cmd_h**, **tag_ctl_par_h**,[4] **tag_dirty_h**,[4] **tag_shared_h**,[4] **tag_valid_h**[4]	Output delay	**Tdd**+0.4 ns	**Tdd**+0.5 ∗**Tcycle**+0.9 ns	**Taod**	**Traod**
data_check_h[2]	Output delay	**Tdd+Tcycle**+0.4 ns	**Tdd**+1.5∗**Tcycle**+0.9 ns	**Tdod**	**Trdod**
addr_cmd_par_h, **cmd_h**, **tag_ctl_par_h**,[4] **tag_dirty_h**,[4] **tag_shared_h**,[4] **tag_valid_h**[4]	Output hold	**Tmdd**	**Tmdd**	**Taoh**	**Traoh**
data_check_h[2]	Output hold	**Tmdd+Tcycle**	**Tmdd+Tcycle**	**Tdoh**	**Trdoh**

[2]Write transaction

[4]Only for write broadcasts and system transactions

Signals in Table 33 are used to control Bcache data transfers. These signals are driven off the CPU clock. The choice of **sys_clk_out** or **ref_clk_in** has no impact on the timing of these signals.

Table 33 Bcache Control Signal Timing

Signal	Specification	Value	Name
Input mode:			
tag_data_h, **tag_data_par_h**, **tag_valid_h**	Input setup	1.1 ns	**Tdsu**
tag_data_h, **tag_data_par_h**, **tag_valid_h**	Input hold	0 ns	**Tdh**
Output mode:			
data_ram_oe_h, **data_ram_we_h**,[1] **tag_ram_oe_h**, **tag_ram_we_h**[1]	Output delay	**Tdd**+0.4 ns	**Taod**
tag_data_h, **tag_data_par_h**, **tag_valid_h**	Output delay	**Tdd**+0.4 ns	**Taod**
data_ram_oe_h, **data_ram_we_h**,[1] **tag_ram_oe_h**, **tag_ram_we_h**[1]	Output hold	**Tmdd**	**Taoh**
tag_data_h, **tag_data_par_h**, **tag_valid_h**	Output hold	**Tmdd**	**Taoh**

[1] Pulse width for this signal is controlled through the BC_CONFIG IPR.

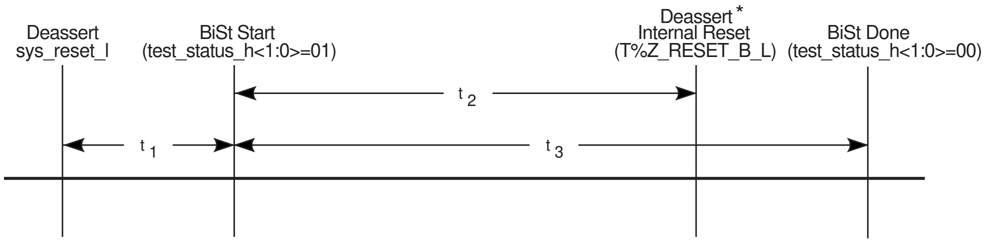
11.4.5 Timing of Test Features

Timing of 21164 testability features depends on the system clock rate and the test port's operating mode. This section provides timing information that may be needed for most common operations.

11.4.6 Icache BiSt Operation Timing

The Icache BiSt is invoked by deasserting the external reset signal **sys_reset_l**. Figure 18 shows the timing between various events relevant to BiSt operations.

Figure 18 BiSt Timing Event–Time Line



The timing for deassertion of internal reset (time t_2, see asterisk) is valid only if an SROM is not present (indicated by keeping signal **srom_present_l** deasserted). If an SROM is present, the SROM load is performed once the BiSt completes. The internal reset signal T%Z_RESET_B_L is extended until the end of the SROM load (Section 11.4.7). In this case, the end of the time line shown in Figure 18 connects to the beginning of the time line shown in Figure 19.

Table 34 and Table 35 list timing shown in Figure 18 for some of the system clock ratios. Time t_1 is measured starting from the rising edge of sysclk following the deassertion of the **sys_reset_l** signal.

Table 34 BiSt Timing for Some System Clock Ratios, Port Mode=Normal (System Cycles)

Sysclk	System Cycles		
Ratio	t_1	t_2	t_3
3	8	22644+2½	22645
4	7	19721+2½	19722
15	7	13291+14½	13292

Table 35 BiSt Timing for Some System Clock Ratios, Port Mode=Normal (CPU Cycles)

Sysclk	CPU Cycles		
Ratio	t_1	t_2	t_3
3	24	67934½	67935
4	28	78886½	78888
15	105	199379½	199380

11.4.7 Automatic SROM Load Timing

The SROM load is triggered by the conclusion of BiSt if **srom_present_l** is asserted. The SROM load occurs at the internal cycle time of approximately 126 CPU cycles for **srom_clk_h**, but the behavior at the pins may shift slightly.

Timing events are shown in Figure 19 and are listed in Table 36 and Table 37.

Figure 19 SROM Load Timing Event–Time Line

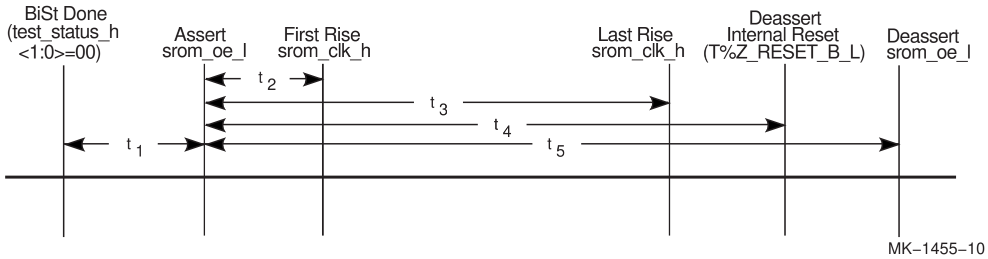


Table 36 SROM Load Timing for Some System Clock Ratios (System Cycles)

Sysclk	System Cycles[1]				
Ratio	t_1	t_2	t_3	t_4	t_5
3	4	22	4408090	4408216+½	4408217
4	3	48	3306099	3306193+2½	3306194
15	3	13	881627	881651+9½	881652

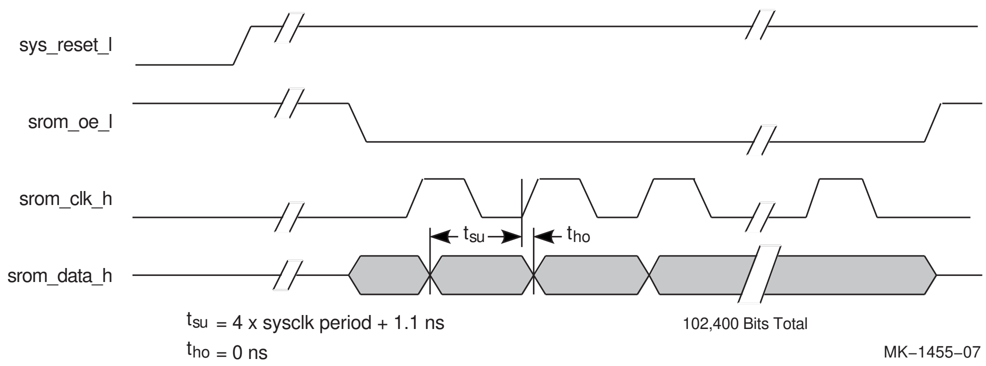
[1]Measured in sysclk cycles, where +*n* refers to an additional *n* CPU cycles.

Table 37 SROM Load Timing for Some System Clock Ratios (CPU Cycles)

Sysclk Ratio	CPU Cycles t_1	t_2	t_3	t_4	t_5
3	12	66	13224270	13224648½	13224651
4	12	192	13224396	13224774½	13224776
15	45	195	13224405	13224774½	13224780

Figure 20 is a timing diagram of an SROM load sequence.

Figure 20 Serial ROM Load Timing



The minimum **srom_clk_h** cycle = (126 − sysclk ratio) ∗ (CPU cycle time).

The maximum **srom_clk_h** to **srom_data_h** delay allowable (in order to meet the required setup time) = [126 − (5 ∗ sysclk ratio)] ∗ (CPU cycle time).

11.4.8 Clock Test Modes

This section describes the 21164 clock test modes.

11.4.9 Normal Mode

When the **clk_mode_h<1:0>** signals are not asserted, the **osc_clk_in_h,l** frequency is divided by 2. This is the normal operational mode of the clock circuitry.

11.4.10 Chip Test Mode

To lower the maximum frequency that the chip manufacturing tester is required to supply, a divide-by-1 mode has been designed into the clock generator circuitry. When the **clk_mode_h<0>** signal is asserted and **clk_mode_h<1>** is not asserted, the clock frequency that is applied to the input clock signals **osc_clk_in_h,l** bypasses the clock divider and is sent to the chip clock driver. This allows the chip internal circuitry to be tested at full speed with a one-half frequency **osc_clk_in_h,l**.

11.4.11 Module Test Mode

When the **clk_mode_h<0>** signal is not asserted and **clk_mode_h<1>** is asserted, the clock frequency that is applied to the input clock signals **osc_clk_in_h,l** is divided by 4 and is sent to the chip clock driver. The digital phase-locked loop (DPLL) continues to keep the onchip **sys_clk_out1_h,l** locked to **ref_clk_in_h** within the normal limits if a **ref_clk_in_h** signal is applied (0 ns to 1 **osc_clk_in_h,l** cycle after **ref_clk_in_h**).

11.4.12 Clock Test Reset Mode

When both the **clk_mode_h<0>** and the **clk_mode_h<1>** signals are asserted, the **sys_clk_out** generator circuit is forced to reset to a known state. This allows the chip manufacturing tester to synchronize the chip to the tester cycle. Table 38 lists the test modes.

Table 38 Test Modes

Mode	clk_mode_h<0>	clk_mode_h<1>
Normal	0	0
Chip test	1	0
Module test	0	1
Clock reset	1	1

11.4.13 IEEE 1149.1 (JTAG) Performance

Table 39 lists the standard mandated performance specifications for the IEEE 1149.1 circuits.

Table 39 IEEE 1149.1 Circuit Performance Specifications

Item	Specification
trst_l is asynchronous. Minimum pulse width.	4 ns
trst_l setup time for deassertion before a transition on **tck_h**.	4 ns
Maximum acceptable **tck_h** clock frequency.	16.6 MHz
tdi_h/tms_h setup time (referenced to **tck_h** rising edge).	4 ns
tdi_h/tms_h hold time (referenced to **tck_h** rising edge).	4 ns
Maximum propagation delay at pin **tdo_h** (referenced to **tck_h** falling edge).	14 ns
Maximum propagation delay at system output pins (referenced to **tck_h** falling edge).	20 ns

11.5 Power Supply Considerations

For correct operation of the 21164, all of the **Vss** pins must be connected to ground and all of the **Vdd** pins must be connected to a 3.3 V ±5% power source. This source voltage should be guaranteed (even under transient conditions) at the 21164 pins, and not just at the PCB edge.

Plus 5 V is not used in the 21164. The voltage difference between the **Vdd** pins and **Vss** pins must never be greater than 3.6 V. If the differential exceeds this limit, the 21164 chip will be damaged.

11.5.1 Decoupling

The effectiveness of decoupling capacitors depends on the amount of inductance placed in series with them. The inductance depends both on the capacitor style (construction) and on the module design. In general, the use of small, high frequency capacitors placed close to the chip package's power and ground pins with very short module etch will give best results. Depending on the user's power supply and power supply distribution system, bulk decoupling may also be required on the module.

Each individual case must be separately analyzed, but generally designers should plan to use at least 6 μF of capacitance. Typically, 40 to 60 small, high frequency 0.1-μF capacitors are placed near the chip's **Vdd** and **Vss** pins. Actually placing the capacitors in the pin field is the best approach. Several tens of μF of bulk decoupling (comprised of tantalum and ceramic capacitors) should be positioned near the 21164 chip.

Use capacitors that are as physically small as possible. Connect the capacitors directly to the 21164 **Vdd** and **Vss** pins by short (0.64 cm [0.25 in] or less) surface etch. The small capacitors generally have better electrical characteristics than the larger units, and will more readily fit close to the IPGA pin field.

11.5.2 Power Supply Sequencing

Although the 21164 uses a 3.3-V (nominal) power source, most of the other logic on the PCB probably requires a 5-V power supply. These 5-V devices can damage the 21164's I/O circuits if the 5-V power source powering the PCB logic and the 3.3-V (**Vdd**) supply feeding the 21164 are not sequenced correctly.

Caution

To avoid damaging the 21164's I/O circuits, the I/O pin voltages must not exceed 4.0 V until the **Vdd** supply is at least 3.0 V or greater.

This rule can be satisfied if the **Vdd** and the 5-V supplies come up together, or if the **Vdd** supply comes up before the 5-V supply is asserted. Bringing the lower voltage up before the higher voltage is the opposite of the way that CMOS systems with multiple power supplies of different voltages are usually sequenced, but it is required for the 21164.

A three-terminal voltage regulator can be used to make 3.3-V **Vdd** from the 5-V supply, provided the output of the regulator (**Vdd**) tracks the 5-V supply with only a small offset. The requirement is that when the 5-V supply reaches 4.0 V, **Vdd** must be 3.0 V or higher. While the 5-V supply is below 4.0 V, **Vdd** can be less than 3.0 V.

All 5-V sources on the 21164's I/O pins should be disabled if the power supply sequencing is such that the 5-V supply will exceed 4.0 V before **Vdd** is at least 3.0 V. The 5-V sources should remain disabled until the **Vdd** power supply is equal to or greater than 3.0 V.

Disabling all 5-V sources can be very difficult because there are so many possible sneak paths. Inputs, for example, on bipolar TTL logic can be a source of current, and will put a voltage across a 21164 I/O pin high enough to violate the (no higher than 4.0 V until there is 3.0 V) rule. TTL outputs are specified to drive a logic one to at least 2.4 V, but usually drive voltages much higher. CMOS logic and CMOS SRAMs usually drive ''full rail'' signals that match the value of the 5-V power supply.

Another concern is parallel (dc) terminations or pull-ups connected between the 21164 and the 5-V supply. The 3.3 V (**Vdd**) supply should be used to power parallel terminations.

Disabling the non-21164 5-V outputs of PCB logic is generally possible, but raises the PCB complexity and can reduce system performance by increasing critical path timing. If the 5-V logic device has an enable pin, circuits (such as power supply supervisor chips) on the PCB can monitor the **Vdd** and 5-V supplies. When the supervision circuit detects that 5.0 V is increasing from zero while the **Vdd** supply is below 3.0 V, the power supply supervisor circuit produces a disable signal to force all PCB logic with 5-V outputs into the high impedance state. This technique will not prevent bipolar TTL inputs from acting as a 5-V source, but it can be used to disable sources such as cache RAM outputs.

12 Thermal Management

This section describes the 21164 thermal management and thermal design considerations.

12.1 Operating Temperature

The 21164 is specified to operate when the temperature at the center of the heat sink (T_c) is 72°C (266 MHz) or 70°C (300 MHz). Temperature (T_c) should be measured at the center of the heat sink (between the two package studs). The GRAFOIL pad is the interface material between the package and the heat sink.

Table 40 lists the values for the center of heat-sink-to-ambient ($\theta_c a$) for the 499-pin grid array. Table 41 shows the allowable T_a (without exceeding T_c) at various airflows.

Note

Digital recommends using the heat sink because it greatly improves the ambient temperature requirement.

Table 40 $\theta_c a$ at Various Airflows

Airflow (linear ft/min)	100	200	400	600	800	1000
Frequency: 266 MHz and 300 MHz						
$\theta_c a$ with heat sink 1 (°C/W)	2.30	1.30	0.70	0.53	0.45	0.41
$\theta_c a$ with heat sink 2 (°C/W)	1.25	0.75	0.48	0.40	0.35	0.32

Table 41 Maximum T_a at Various Airflows

Airflow (linear ft/min)	100	200	400	600	800	1000
Frequency: 266 MHz, Power: 46 W @Vdd = 3.3 V						
T_a with heat sink 1 (°C)	—	—	39.8	47.6	51.3	53.2
T_a with heat sink 2 (°C)	14.5	37.5	49.9	53.6	55.9	57.3
Frequency: 300 MHz, Power: 51 W @Vdd = 3.3 V						
T_a with heat sink 1 (°C)	—	—	34.3	43.0	47.1	49.1
T_a with heat sink 2 (°C)	—	31.8	45.5	49.6	52.2	53.7

12.2 Heat Sink Specifications

Two heat sinks are specified. Heat sink type 1 mounting holes are in line with the cooling fins. Heat sink type 2 mounting holes are rotated 90° from the cooling fins. The heat sink composition is aluminum alloy 6063. Type 1 heat sink is shown in Figure 21, and type 2 heat sink is shown in Figure 22, along with their approximate dimensions.

Figure 21 Type 1 Heat Sink

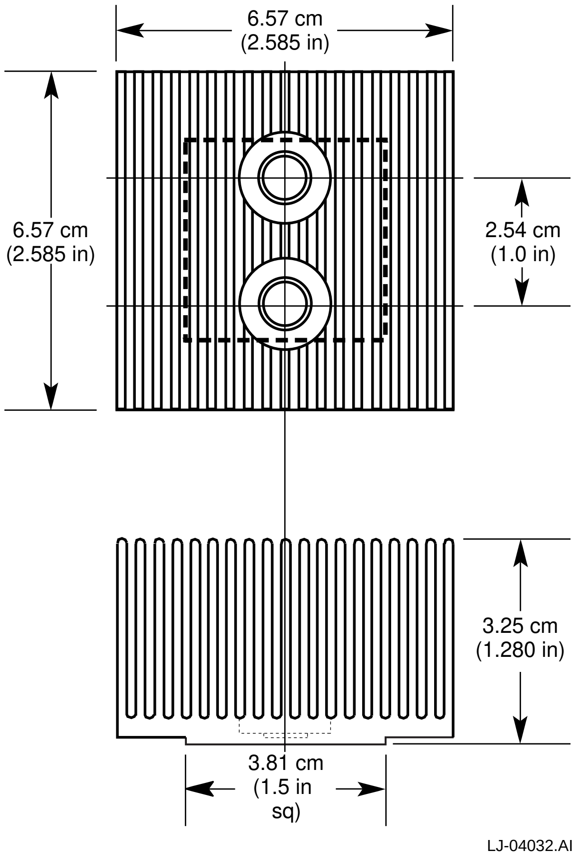
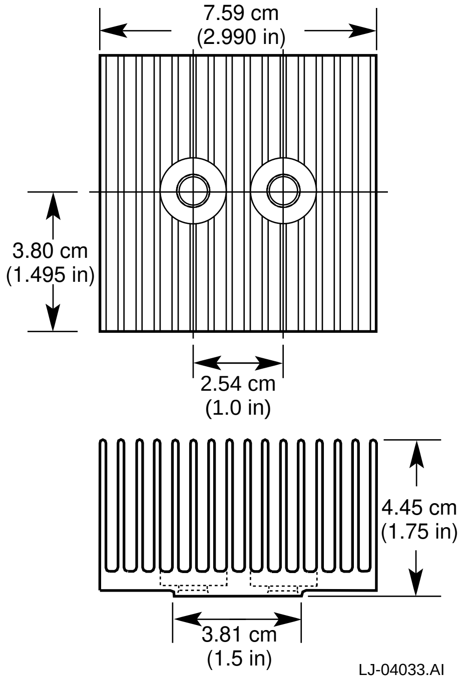


Figure 22 Type 2 Heat Sink



12.3 Thermal Design Considerations

Follow these guidelines for printed circuit board (PCB) component placement:

- Orient the 21164 on the PCB with the heat sink fins aligned with the airflow direction.
- Avoid preheating ambient air. Place the 21164 on the PCB so that inlet air is not preheated by any other PCB components.
- Do not place other high power devices in the vicinity of the 21164.
- Do not restrict the airflow across the 21164 heat sink. Placement of other devices must allow for maximum system airflow in order to maximize the performance of the heat sink.

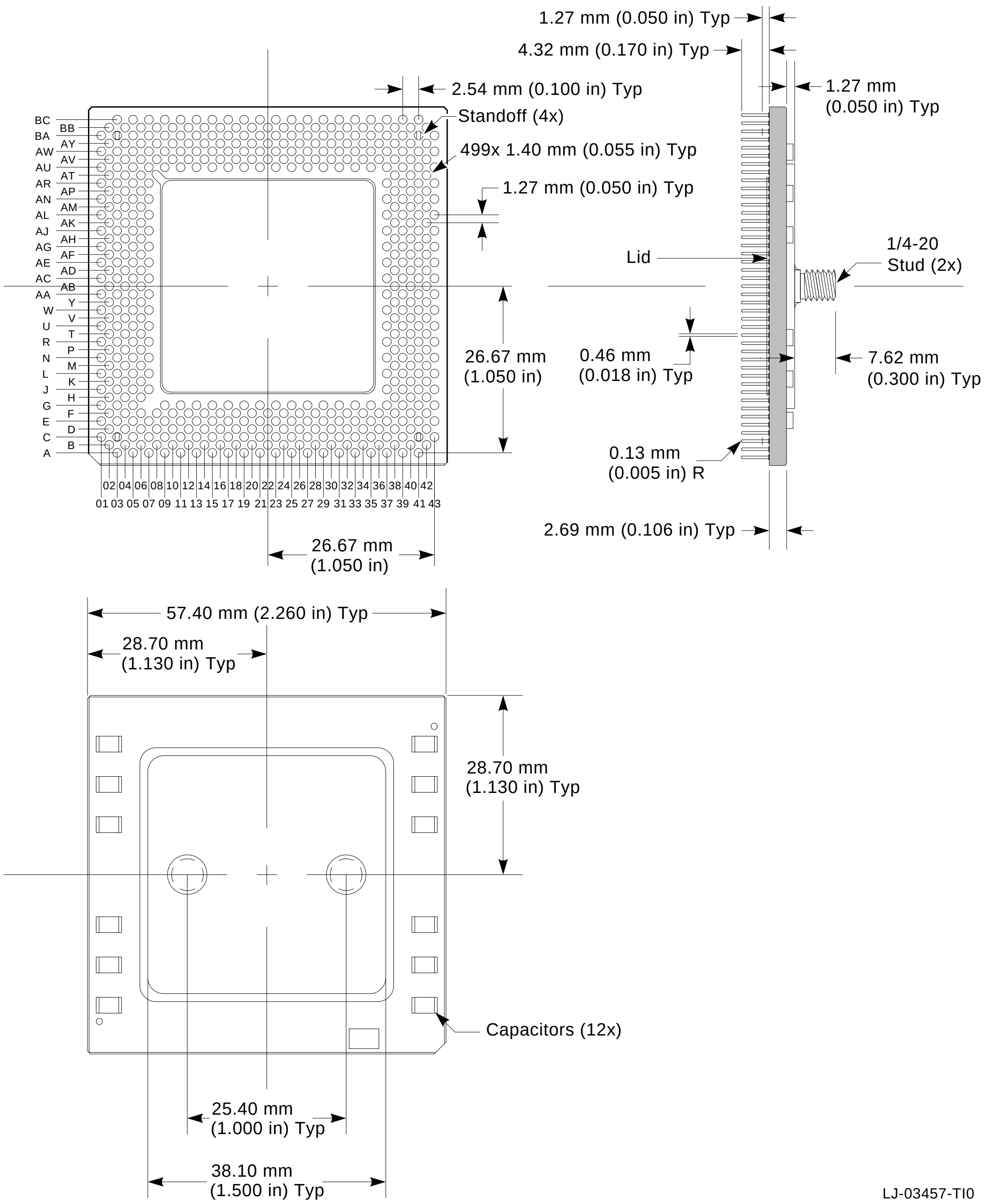
13 Mechanical Specifications

This section shows the 21164 mechanical package dimensions without a heat sink. For heat sink information and dimensions, refer to Section 12.

Package Dimensions

Figure 23 shows the package physical dimensions without a heat sink.

Figure 23 Package Dimensions



Technical Support and Ordering Information

Technical Support

If you need technical support or help deciding which literature best meets your needs, call the Digital Semiconductor Information Line:

United States and Canada	**1–800–332–2717**
TTY (United States only)	**1–800–332–2515**
Outside North America	**+1–508–568–6868**

Ordering Digital Semiconductor Products

To order the Alpha 21164 microprocessor and evaluation boards, contact your local distributor.

You can order the following semiconductor products from Digital:

Product	Order Number
Alpha 21164 300 MHz Microprocessor	21164–300MHZ
Alpha 21164 266 MHz Microprocessor	21164–266MHZ
Alpha 21164 Microprocessor Evaluation Board 266 MHz Kit (Supports OSF/1, OpenVMS, and Windows NT operating systems.)	21A04–01

Ordering Digital Semiconductor Sample Kits

To order an Alpha 21164 Microprocessor Sample Kit, which contains one Alpha 21164 microprocessor, one heat sink, and supporting documentation, call **1–800–DIGITAL**. You will need a purchase order number or credit card to order the following products:

Product	Order Number
Alpha 21164–266 Sample Kit	21164–SA

Ordering Associated Literature

The following table lists some of the available Digital Semiconductor literature. For a complete list, contact the Digital Semiconductor Information Line.

Title	Order Number
Alpha Architecture Reference Manual[1]	EY–L520E–DP–YCH
Alpha AXP Architecture Handbook	EC–QD2KA–TE
Alpha 21164 Microprocessor Hardware Reference Manual	EC–QAEQB–TE
Alpha 21164 Microprocessor Product Brief	EC–QAENB–TE
Alpha 21164 Evaluation Board Read Me First	EC–QD2VA–TE
Alpha 21164 Evaluation Board Product Brief	EC–QCZZB–TE
Alpha 21164 Evaluation Board User's Guide	EC–QD2UA–TE
DECchip 21171 Core Logic Chipset Product Brief	EC–QC3EB–TE
DECchip 21171 Core Logic Chipset Technical Reference Manual	EC–QE18A–TE
Answers to Common Questions about PALcode for Alpha AXP Systems	EC–N0647–72
PALcode for Alpha AXP Microprocessors System Design Guide	EC–N0660–72
Alpha Microprocessor Evaluation Board System Configuration and Windows NT 3.5 Installation Guide	EC–QD2YA–TE
SPICE Models for Alpha Microprocessors and Peripheral Chips: An Application Note	EC–QA4XB–TE
Alpha Microprocessors SROM Mini-Debugger User's Guide	EC–QHUXA–TE
Alpha Microprocessors Evaluation Board Debug Monitor User's Guide	EC–QHUVA–TE
Alpha Microprocessors Evaluation Board Software Design Tools User's Guide	EC–QHUWA–TE

[1] To order and purchase the *Alpha Architecture Reference Manual*, call **1–800–DIGITAL** from the U.S. or Canada, or contact your local Digital office, or technical or reference bookstore where Digital Press books are distributed by Prentice Hall.

Ordering Associated Third-Party Literature

You can order the following third-party literature directly from the vendor:

Title	Vendor
PCI System Design Guide	PCI Special Interest Group N/S HH3–15A 5200 N.E. Elam Young Pkwy Hillsboro, Oregon 97124–6497 1–503–696–2000
PCI Local Bus Specification, Rev 2.0	See previous entry.
IEEE Standard 754, *Standard for Binary Floating-Point Arithmetic*	IEEE Service Center 445 Hoes Lane P.O. Box 1331 Piscataway, NJ 08855–1331 1–800–678–IEEE (U.S. and Canada) 908–562–3805 (Outside U.S. and Canada)
IEEE Standard 1149.1, *A Test Access Port and Boundary Scan Architecture*	See previous entry.