DEC Alpha

DIGITAL ANNOUNCES THE WORLD'S FASTEST MICROPROCESSOR

...New Alpha open computing architecture and new business practices lead way into 21st century computing

Digital Equipment Corporation today unveiled the world's fastest microprocessor, code-named Alpha, marking a breakthrough in open chip architecture. The new generation of computer technology was revealed to European journalists at Digital's $200 million state-of-the-art facility at South Queensferry, Scotland, one of two plants worldwide which will manufacture the chip.

"Alpha is a totally new, open computing environment that will give users the benefits of advanced 21st century computing," said Pier Carlo Falotti, President of Digital Europe. "It will create a new industry standard - one which supports multiple operating systems and will increase in performance by a factor of 1,000 over its anticipated 25-year life".

The new, open architecture will form the 'heart' of systems that span from the palmtop to the supercomputer. Users will have the flexibility to deploy current applications on popular operating environments at peak performance, beginning with OSF/1 and VMS. Digital also announced that it planned to license the Alpha architecture to other semiconductor manufacturers, and actively market the Alpha technology at all levels of integration - chip, board, and system.

ALPHA - A BREAKTHROUGH IN CHIP TECHNOLOGY

The first product which results from the Alpha programme will be Digital's 21064 RISC microprocessor. The 64-bit chip has up to four billion times more address space than 32-bit implementations from IBM, Hewlett Packard, and Sun. Its clock speed is twice as fast as any competitor, roughly the same peak performance as a Cray-1 supercomputer for the cost of today's workstation. It is priced at $3375 each in units of 1 to 100; $1650 in units of 101 to 1,000; and $1559 for over 1,000.

Alpha's delivery of affordable supercomputing power to the desktop will open a whole new area of power-hungry applications once handled only by supercomputers or large mainframes. Seismic data analysis, financial closing, molecular modelling, engineering design and many others will be able to run on Alpha at a fraction of today's mainframe costs. Alpha's power will provide the colour and graphics for emerging personal use applications that are straining the limits of today's PCs, such as voice and video,visualisation systems, imaging, and artificial intelligence.

"Alpha places Digital ahead, not only of current high-end chips from our competitors, but of their next generation as well. The beauty of the technology is that users will have the freedom to choose the best applications today in the knowledge that all their existing investments, will be protected," said Peter Graham, Alpha Business Manager.

NEW OPEN BUSINESS PRACTICES

With Alpha set to become the new standard in computing, Digital will license the new technology to third party vendors. Cray Research Inc. has already announced that Alpha will form the basis of its first-generation massively parallel processor supercomputer, and Kubota Inc. will use the chip in its high-performance graphics workstation, available in late 1992. "The Alpha chip delivers leading edge single-chip functionality and performance,' said Cray Research Chairman, John Rollwagen, at the time of its announcement.

Digital is offering a comprehensive support programme to help companies migrate their applications to Digital's Alpha platform. Extensive porting activities are already underway with many application software vendors. Over the next months, Digital will announce additional alliances and partnerships that will highlight third-party Alpha support.

Such alliances and partnerships will ensure the broadest use of the architecture with the widest possible range of operating environments. The company will partner with software vendors and provide an extensive portfolio of applications on Alpha, and license its operating systems (including DEC OSF/1 and VMS), compilers, and layered software products.

"Licensing Alpha is in line with Digital's commitment to deliver the Open Advantage", said Falotti. "It will accelerate Alpha's acceptance as a new industry standard, which in turn will generate revenue for applications developers and broaden the range of products available. Thus third parties have a stake in Digital's future, and vice versa, and the ultimate beneficiary of this open approach is the customer".

COMPLETE CUSTOMER AND OEM SUPPORT FOR ALPHA

Customers will continue to buy leading UNIX and VMS systems from Digital knowing that they have a clear migration path to 21st century computing. Digital will support Alpha customers and vendors with services ranging from consulting to education and training, client/server systems management, product design, and integration and migration services - all designed to make users successful in planning, designing, implementing, and managing new Alpha computing environments.

Concluded Falotti:"Alpha is the distillation of expertise Digital has gained from 35 years of innovative architectural design and chip technology. With its unsurpassed power and ability to run different operating systems, Alpha is one of the most significant advances in the computer industry for many years.

"In the next five years only a handful of computer manufacturers will be able to design and build microprocessors, and enjoy the inherent advantage in performance and time to market. The rest will use off-the-shelf components and will compete on the basis of services. With Alpha, Digital will be one of those few, and first among equals".

ALPHA AT A GLANCE

The Alpha programme consists of two elements:

- the first 21st century computing architecture - a totally new, 64-bit reduced instruction set computing (RISC) architecture designed to be fast, reliable and open.
- a single chip implementation of this new architecture. The 21064 chip has double the clock speed and up to four billion times more address space than competitors' 32-bit implementations, with roughly the same peak performance as a Cray-1 supercomputer.

Why is Alpha an industry first ?

Alpha is the first microprocessor architecture to combine the following:

- it is designed to be scalable by over 1,000 times over its 25-year lifetime to meet user demands for more power.
- its open technology is able to run multiple operating systems.
- its design supports both single processor and massively parallel systems.
- it will be licensed at every level of integration - chip, board and system: the most compelling example of open business practices to date.
- it will be implemented in successive generations of 64-bit micro processors spanning from palmtop to supercomputer.

ALPHA KEY FACTS

Open Advantage

- Alpha is another proof of Digital's commitment to openness as embodied in its Open Advantage. The chip will create a new industry standard and is a world first - a full RISC implementation with no operating system or language bias.
- Alpha is the first microprocessor with the ability to run different operating systems. It is capable of running not just VMS software but also OSF/1 and a dozen other operating systems and languages.
- The OSF/1 operating system can be run on Alpha, MIPS and Intel platforms, offering varying price points and performance to customers. OSF/1 and Open VMS will be the first operating systems available on Alpha.

Open Business Practices

Digital has lead the way in formulating standards, licensing software, supporting other vendors' equipment and applications, and working closely with third parties.

In line with a set of open business practices, Digital will now license the new technology to third party vendors. Cray Research - the world's leading supercomputing manufacturer - has already announced its choice of Alpha for its first-generation massively parallel processing system.

Digital will license Alpha at all levels of integration - chip, board and system - to other computer companies and to original equipment manufacturers. Such alliances will ensure the broadest use of the architecture with the widest possible range of operating environments, and guarantee that it will reach the market at the most attractive price.

Digital will partner with software vendors and application developers to deliver a broad portfolio of applications, and license its operating systems (including DEC OSF/1 and VMS), compilers, and layered software products.

Outstanding performance

Alpha is the world's fastest single chip microprocessor, and the first 64-bit RISC chip from a major computer vendor. Alpha runs at twice the speed of HP's latest chip, and is thousands of times more cost-effective than IBM's mainframes. Digital believes it will outlive, outscale and out perform all current architectures in the industry.

The Alpha architecture will form the heart of Digital's strategy for 21st century computing, going beyond the current RISC curve. Its 64-bit technology has 4 billion times the address space of a 32-bit system, and should last at least 25 years.

It represents the industry's first ever simultaneous leap forward in power, size, compatibility and industry standards.

Alpha will deliver all the power of a Cray-1 supercomputer for the price of a workstation, able to run power-hungry applications at a fraction of today's mainframe costs. Over the next 25 years it will scale up to 1,000 times, perhaps even 10,000 times, as user demands increase.

Alpha systems will be industry-leading in price/performance and performance, providing the assurance that users will never have to upgrade again in their career lifetime.

More choice

The Alpha architecture will complement Digital's support of other industry standards, including the ACE initiative. OSF/1 on Alpha will be compatible with OSF/1 on ACE.

Products coming out of the Alpha programme will coexist with - not replace - Digital's current product range of VAX and UNIX systems.

Alpha powerfully underlines Digital's operating philosophy: one architecture scalable from palmtop to mainframe, from the simplest of invoicing tasks to the most complex of scientific challenges.

Investment Protection

Digital will continue to protect its customers' investments in IT. Alpha's open architecture and system design provides a clear migration path, giving investment protection for data, applications, interfaces and peripherals.

Digital's eight million VAX customers, running over 8,000 applications, have a clear upgrade path into the 21st century with minimum inconvenience.

Migrating applications to OSF/1 and VMS on ALPHA is easy. All OSF/1 and VAX/VMS users can port and run the same software with the same data storage applications.

Enabling Technology

Alpha is an enabling technology; it is the 'engine' that will drive applications and solutions that were previously unachievable or too costly.

The Alpha microprocessor is a new generation of technology which will open new opportunities for interacting with computers at work and home, up to and well beyond the year 2000.

Alpha's power will provide the colour and graphics for emerging personal use applications that are straining the limits of today's PCs, such as voice and video, visualisation systems and imaging.

Made in Europe

Alpha will be manufactured both in Hudson (Massachusetts) and at the South Queensferry plant in Scotland from spring 1992.

Digital is the only company with this type of manufacturing capability in Europe. It has invested an extra $54 million to be able to produce the Alpha chip in South Queensferry on top of its $137 million in the plant to date.

Complete support

Digital is offering a comprehensive support programme to help companies migrate any vendor's applications to Alpha. Extensive porting activities are already underway with many application software vendors.

Digital's full range of support services include consulting, education and training, client/server systems management, and integration and migration services. All are designed to make users successful in planning, designing, implementing, and managing new Alpha.

Digital will support vendors during product design, implementation, and subsequent marketing. Digital can also become the service provider for the vendor's Alpha-based products through different distribution channels, OEMs, VARs, or direct.

The first of 30 Alpha Upgrade Centres have already been established in the USA, Europe and the Asia Rim, staffed by software support personnel with expertise in VMS, ULTRIX and DEC OSF/1.

ANALYST OPINION OF ALPHA

"In the constant 'leap-frog' race Digital has definitely taken a giant step ahead of the competition with Alpha. The R&D costs of developing a new chip architecture are incredibly high, out of reach of many smaller vendors. This chip offers them a broad based technology with operating systems that are generally available, which makes it very attractive. Alpha will also provide the very high power required for countless multimedia applications for the next several generations."

Lars Mieritz, Technology Investment Strategies Corporation Ltd (UK)

"Alpha represents a new performance generation, both in terms of the type of chip it is and the very high speeds at which Digital will run the chip. It is definitely the most powerful chip available in the whole world today. At the same time, Digital is entering the market as an original equipment manufacturer (OEM) chip supplier, and will be able to extend its external industry importance by making the chips available to other companies. If the Alpha programme continues to develop partnerships such as that with Cray Research, Digital will re-assert itself as a full range supplier and win back its reputation for being a leading edge component designer."

Martin Hingley, International Data Corporation (UK)

"Alpha breaks new ground in chip design. This technology should place Digital in front of Sun and HP's current high-end chips and in front of their next generation as well."

Harry William, Sanford C. Bernstein (US)

"If the figures are correct, Digital has moved in one year from last to first in hardware technology".

Marc Schulman, UBS Securities (US)

ALPHA - A 21ST CENTURY COMPUTING ARCHITECTURE

Key to every computing solution is a set of rules about how data is addressed and instructions are handled. It's called a "computing architecture".

To the extent that a computing architecture is robust and long-lived-spanning a large range of systems, many kinds of applications, and multiple generations of technology - it delivers important benefits: Transitions to new systems and new technologies are smooth and non-disruptive; Investments in hardware, software, and data are protected; Operational and support requirements are simplified and the overall cost of ownership is lowered.

DOES THE WORLD NEED ANOTHER COMPUTING ARCHITECTURE ?

Thirty years of computing have shown that today's mainframe is tomorrow's notebook. Customers need a range of computing solutions to address variously "sized" problems. A computing architecture that does not scale well across a broader range of systems is soon limited in its usefulness. At a minimum, an architecture for the 21st century must be able, from the beginning, to be implemented on a single chip on the low end - and to support a very large, even massively parallel multiprocessing system on the high end.

While systems based on both CISC and RISC 32-bit architectures will continue to deliver leading-edge performance for years to come, they will not be able to keep pace with improvements in memory technology to meet the demands of the "supercomputing-like" applications of the 21st century, such as visualisation, very large databases, imaging, multimedia, simulation and modelling. Quite simply, the 32-bit systems are going to run out of address space.

64-BIT ARCHITECTURE

The answer will be a computing architecture which will be a full-speed 64-bit computing architecture. The development of this technology will open a whole new area of applications for computers. The last ten years have seen tremendous growth in demand for power from CAD/CAM and simulation packages, massive databases, multimedia, electronic publishing systems, econometric modelling software, and so on. All of these packages have increasingly used memory-hungry colour and graphics to improve their user interfaces.

Moving from a 32-bit to 64-bit architecture should provide a comfortable 50 years of growth. The move from a 32-bit to 64-bit is not just a doubling, it provides four billion times the address space - and the architectural underpinnings and performance to last 25 years.

A 25 YEAR DESIGN HORIZON

Over the last ten years, computing performance has improved by a factor of 100. Given the ever-accelerating rate of technical advancement and demand for performance, it is very likely that a 25-year architecture will need to be scalable by a factor of 1,000.

This means that a 21st century architecture must be able to take advantage of performance improvements in all three dimensions of performance: CPU clock speed, multi-issue instruction (superscalar), and multiple processors, including massively parallel processing.

For Digital's customers, the Alpha architecture will do just that. It represents a simultaneous leap forward in power, size, compatibility and industry standards. It combines supercomputing levels of power with the ability to scale up by at least 1,000 times as user needs increase, and designed to be the 'heart' of any size of machine from the palmtop to the supercomputer.

OPEN TECHNOLOGY

Perhaps the most dramatic change to come will be the idea of an open computing architecture - one that can be implemented by many different vendors at all levels of integration. Unlike all past architectures, a computing architecture for the 21st century will not be targeted for, nor show any bias towards, any particular operating system, language, or style of computing.

As the computing industry moves into the 21st century, computer vendors and users alike are experiencing both the benefits and the necessities of 'multi-vendor' solutions. To be open, an architecture must be designed to allow customers to mix and match chips, systems, compilers, and software from multiple sources to meet their needs.

Again, this 'openness' is built into the Alpha architecture. By offering the OSF/1 operating system on Alpha, MIPS and Intel platforms, users will have the choice of varying price points and performance. Most applications can migrate to Alpha very easily, and simply need to be recompiled into the Alpha code.

THE IMPACT OF MICROPROCESSOR TECHNOLOGY

One of the key competitive factors in the computer industry is microprocessor technology. In the past few years, the power of the microprocessor has soared, so much so that computers built using multiple off-the-shelf microprocessors are competing with traditional mainframes and supercomputers at a fraction of the cost.

Microprocessors have set the entire industry on a trend towards standard hardware platforms. But the cost of developing new microchip technologies - and building a factory to make them - is several hundred million dollars. It would cost $500 million to set up South Queensferry today. By 1995, the entry bar will have been raised to $1 billion.

Over the next five years this will cause a convergence on a few microprocessor architectures. Only a handful of computer manufacturers will be able to design and build their own. They will enjoy an advantage in performance and time to market, while other will use off-the-shelf components and will compete on the basis of services.

In short, only those companies with control over their own technological destinies will survive.

ALPHA - TECHNICAL CHARACTERISTICS

Alpha is a multiple-instruction issue, pipelined, 64-bit, load/store, reduced instruction set architecture.

It's the world's fastest IEEE compatible floating point chip.

It is a dual-instruction processor, meaning that it can handle two instructions at once.

It is a CMOS (Complimentary Metal Oxide Semiconductor) with 1.7m of transistors. Minimum feature size is 0.75 microns, transistor channel length is 0.5 microns and the chip operates at 3.3 volts.

It handles 30 watts of power with 64-bit virtual and physical addresses and 64-bit integers and floating points, with no operating system or language bias and four billion times the addressable space of existing chips.

With clock rates of up to 200MHz, the chip is capable of delivering 400 peak MIPs and 200 peak MFLOPS - over twice most of Digital's competitors.

With a total transistor count of 1.68 million devices, the chip is a complete CPU, including full integer and floating-point execution units. These units, together with related addressing and branching units, are fully pipelined, and each is capable of launching a new operation every cycle.

The chip includes two high speed memory caches. An eight Kbyte instruction cache provides two full 32-bit instructions per clock cycle to the instruction dispatch unit, and an eight Kbyte data cache can provide a 64-bit data access during each cycle. The resulting cache bandwidth of 3.2 Gigabytes/second far exceeds what could be accomplished if these cache units were not fully integrated.

Initially, Alpha systems will be able to run OSF/1 and VMS - there is no bias towards a particular operating system or programming language in the architecture.

ALPHA PERFORMANCE & COMPETITIVE POSITIONING

Vendor	Digital	MIPS	Sun/TI	IBM	HP	Intel	Motorola
Device	21064	R4000	Viking	RIOS	PA-4	i860XP	88110
Max.Freq Internal	200 Mhz	100 Mhz	50 Mhz	50 Mhz	66 Mhz	50 Mhz	50 Mhz
No.Chips Required	1	1	1	7-9	2	1	1
Peak MIPS	400	100	150	200	132	150	150
Peak MFLOPS	200	50	50	100*	132*	100*	100
Base Arch Design	64-bit	64-bit	32-bit	32-bit	32-bit	32-bit	32-bit
Samples Available	Now	Now	N/A	N/A	N/A	Now	Mid-1992

* = combined fmul/fadd

Editor's Notes:

Digital's CMOS technology runs faster than any other semiconductor manufacturer (including Intel, Motorola, IBM, TI, Cypress, NEC and Toshiba). CMOS-4 is the fourth generation of CMOS chip from Digital.

Previous CMOS generations have delivered the world's fastest CISC microprocessors into Digital's products (for example VAX 6000 Model 500 with CMOS-3 chips operating at 62.5 MHz in 1990).

CMOS-4 continues this world leading position in the VAX 6000 Model 600 with the 83MHz chip. This same technology is used to manufacture the Alpha CPU and a wide range of peripheral chips.

BUSINESS/INDUSTRY APPLICATIONS

Alpha will enable users to compete effectively on a global scale, significantly reduce operational costs, communicate and trade with other businesses at unprecedented speed and take advantage of extensive information resources to create and thrive within new markets.

The high performance of Alpha will act as a catalyst in the development of new "intuitive" user interfaces and applications which can keep pace with the speed of the working environment:

Supercomputer Applications

Until recently, supercomputers had been confined to the laboratories of the scientific and research community. Today, increasingly, they are found in a variety of commercial applications - econometric forecasting, designing aircraft, simulating car crashes, modelling molecular chains for drug development, studying the stresses on artificial implants, and many others.

Alpha will put the power required for these computer aided design and engineering applications onto the desktop - at prices equivalent to today's workstation. In essence, there will be a paradigm shift from what was once only conceivable on a supercomputer to a wide-ranging business environment.

Voice Interactive Computing

Digital predicts that by 1995 users will be able to talk to - and be understood by - their computers. This could mark the beginning of the end for the keyboard. DIDDLY*, for example, is a commercial system developed by Digital which enables the user to give commands by voice - and to hear the computer talk back. Hospital operating theatres are already testing its ability to allow surgeons to call up a variety of data while operating. Other potential applications could include voice interactive navigation between a pilot and the control tower computer, route finders for cars and buses, and television shopping for the home user. The list is endless.

* Digital Integrated Distributed Data Library

Multimedia and Video Communications

The ability to combine image, text, graphics and voice on the desktop is one ideally suited to Alpha's cost-effective performance. Jabberwocky is a prototype system from Digital that combines phone, computer and video systems to provide such applications as 'video-conferencing' and 'video mail.' With Alpha, users will be able to communicate on-line with clear and instaneous video transmission, while exchanging text and graphics.

"Active Badges"

"Active Badges" systems use infra red beams to track wearers through a building. The badge's signal enables a central computer system, or other colleagues, to keep track of their whereabouts. Telephone calls can be transferred, security doors opened and computers logged on and off automatically. Again, Alpha will provide the cost-effective power to bring this technology to the wider marketplace within the next few years.

Virtual Reality

Virtual reality is the ability to experience the world of the computer from "within." Estate agents for example, will be able to "take" prospective buyers from room to room, getting a feel for the condition and dimensions of a house for sale. For training purposes, learner drivers will be able to go out on the road without leaving the comfort of their home and in the world of entertainment, who knows, users could get in a good round of golf without even leaving the office !

Pen Based Computing

Alpha will increase the power of systems that recognise handwriting and freehand drawing instead of keyboard commands. With Alpha, writing and graphics will be reproduced instantly and not subject to current delay times. Furthermore, users will be able to write or draw directly onto the screen instead of onto a special tablet as used now.

ALPHA - DID YOU KNOW ?

- The Alpha chip, which is barely a quarter of an inch square, contains nearly two million transistors.
- To make each individual transistor on an Alpha chip visible to the naked eye, the chip would have to be enlarged to the size of Switzerland.
- With a peak speed of 400 million instructions/second, the Alpha chip can launch two instructions in the time it takes light to travel across a room.
- A single Alpha chip has roughly the same peak performance as a Cray-1 system, a small supercomputer.
- The power which Alpha provides on a single chip the size of a fingernail would have occupied a large room as little as 10 years ago.
- In 1975 it took Concorde approx. 3 hours to fly from London to Washington. If Concorde had achieved a comparable improvement in speed as Digital has from its first VAX to today's Alpha, the flight would take less than two minutes !
- By the year 2000, Digital predicts that 80 % of businesses in Europe will be using Alpha technology in some shape or form.

SOUTH QUEENSFERRY - THE WORLD'S MOST ADVANCED SEMICONDUCTOR PLANT

The $200 million facility at South Queensferry - officially opened in September 1990 - is one of the most advanced semiconductor plants in the world and Digital's second largest investment outside the USA to date.

Digital is committed to the European market, which now accounts for 40 % of the corporation's worldwide revenues.

Digital is the only company with this type of manufacturing capability in Europe, and the only computer manufacturer in the UK able to work from the raw silicon, through design and production, to the finished computer.

Digital has invested over $50 million to prepare South Queensferry for worldwide production of the Alpha chip, from spring 1992.

Digital's investment in South Queensferry has created 450 new jobs. In the last year alone a record $6 million has been spent on staff training.

The wafers at South Queensferry are manufactured in a Class I clean room, where the air is kept a thousand times cleaner than an operating theatre and a million times cleaner than an ordinary office. Working at one micron tolerances, the smallest impurity - even a fleck of dust one hundredth the diametre of a human hair - could short circuit the wafer.

Each day, two hundred thousand gallons of water - one million times cleaner than tap water - are used in the washing and fabrication process.

The air supply goes through a series of temperature controls and filtration processes before being fed into the clean room. After use, it is put through a scrubbing process and discharged - as clean as when it came in.

digital™

DECChip™ 21064–AA RISC Microprocessor
Preliminary Data Sheet

Digital Equipment Corporation
Maynard, Massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation April 29, 1992.

All Rights Reserved.
Printed in U.S.A.

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

This document was prepared using VAX DOCUMENT, Version 2.0.

Contents

Chapter 3 Privileged Architecture Library Code

Chapter 4 External Interface

Chapter 5 DC Characteristics

Chapter 6 AC Characteristics

Chapter 7 Package Information

Chapter 8 Pinout

Figures

Tables

Preface

This specification covers the following topics:

- Introduct to terminology and conventions
- An overview of the micro-architecture
- 21064 implimentation of the Privileged Architecture Library Code
- Chip external interface
- Electrical characteristics
- Pinout and Packaging

Chapter 1
Introduction

1.1 Scope

This document describes the 21064-AA RISC CPU microprocessor. The 21064-AA is the first of a family of microprocessors that implement the Digital Equipment Corporation Alpha architecture. This document describes features specific to the 21064 implementation of the architecture. It does not describe the complete details of the implementation nor the Alpha architecture.

1.2 21064 Features Overview

The 21064 microprocessor is a CMOS-4 (.75 micron) super-scalar super-pipelined implementation of the Alpha architecture. It will become the basis of the first family of Alpha products. The 21064 is designed to meet the requirements of a wide variety of systems, ranging from uniprocessor workstations to midrange multiprocessors. To achieve this goal, the CPU enforces as little policy as possible, e.g. it does not enforce a particular cache coherence scheme. The 21064 attempts to spread fairly the design compromises over the range of user requirements. The design balances the cost goals of the low-end workstation with performance goals of the mid-range multiprocessors.

Features Overview:

- Alpha instructions to support byte, word, longword, quadword, DEC F_floating, G_floating and IEEE S_floating and T_floating data types. Limited support is provided for DEC D_ floating operations. 21064 implements the architecturally optional instructions: FETCH and FETCH_M.
- Demand paged memory management unit which in conjunction with properly written PALcode fully implements the Alpha memory management architecture. The translation buffer can be used with alternative PALcode to implement a different page table structure.
  - — On-chip 12-entry I-stream TB with 8 entries for 8K-byte pages and 4 entries for 4M byte pages.

— 32-entry D-stream TB with each entry able to map 8K, 64K, 512K, or 4M byte pages.

- World class performance. At its nominal frequency the 21064 achieves a 6.6ns cycle time. Cycle times of 5ns will also be possible.
- Low average cycles per instructions (CPI). The 21064 CPU can issue two Alpha instructions in a single cycle, thereby minimizing the average CPI. Branch history tables are also used to minimize the branch latency, further reducing the average CPI.
- On-chip high-throughput floating point unit, capable of executing both DEC and IEEE floating point data types.
- On-chip 8K-byte data cache and an 8K-byte physical instruction cache with ASN support.
- On-chip write buffer with four 32-byte entries.
- On-chip performance counters to measure and analyze CPU and system performance.
- Bus interface unit, which contains logic to directly access external cache RAMs without CPU module action. The size and access time of the external cache is programmable.
- An instruction cache diagnostic interface to support chip and module level testing.
- An internal clock generator which generates both a high-speed clock needed by the 21064. and a pair of system clocks for use by the CPU module.
- The 21064 is packaged in a 431 pin (24 x 24, 100 mil pin pitch) PGA package. The heat sink is separable and application specific.

1.3 Terminology and Conventions

1.3.1 Definitions

This specification is the description of the 21064-AA RISC microprocessor. The full designation will be used where appropriate. 21064, CPU and chip will also be used when referring to the 21064-AA.

1.3.2 Numbering

All numbers are decimal unless otherwise indicated. Where there is ambiguity, numbers other than decimal are indicated with the name of the base following the number in parentheses; for example, FF(hex).

1.3.3 UNPREDICTABLE And UNDEFINED

Throughout this specification, the terms UNPREDICTABLE and UNDEFINED are used. Their meanings are quite different and must be carefully distinguished. One key difference is that only privileged software (that is, software running in kernel mode) may trigger UNDEFINED operations, whereas either privileged or unprivileged software may trigger UNPREDICTABLE results or occurrences. A second key difference is that UNPREDICTABLE results and occurrences do not disrupt the basic operation of the processor; the processor continues to execute instructions in its normal manner. In contrast, UNDEFINED operation may halt the processor or cause it to lose information.

A result specified as UNPREDICTABLE may acquire an arbitrary value subject to a few constraints. Such a result may be an arbitrary function of the input operands or of any state information that is accessible to the process in its current access mode. UNPREDICTABLE results may be unchanged from their previous values. UNPREDICTABLE results must not be security holes. Specifically, UNPREDICTABLE results must not do any of the following:

- Depend on or be a function of the contents of memory locations or registers which are inaccessible to the current process in the current access mode
- Write or modify the contents of memory locations or registers to which the current process in the current access mode does not have access
- Halt or hang the system or any of its components

For example, a security hole would exist if some UNPREDICTABLE result depended on the value of a register in another process, on the contents of processor temporary registers left behind by some previously running process, or on a sequence of actions of different processes.

An occurrence specified as UNPREDICTABLE may happen or not based on an arbitrary choice function. The choice function is subject to the same constraints as are UNPREDICTABLE results and, in particular, must not constitute a security hole.

Results or occurrences specified as UNPREDICTABLE may vary from moment to moment, implementation to implementation, and instruction to instruction within implementations. Software can never depend on results specified as UNPREDICTABLE

Operations specified as UNDEFINED may vary from moment to moment, implementation to implementation, and instruction to instruction within implementations. The operation may vary in effect from nothing, to stopping system operation. UNDEFINED operations must not cause the processor to hang, i.e., reach a state a state from which there is no transition to a normal state in which the machine executes instructions and not be halted. Only privileged software (that is, software running in kernel mode) is allowed to trigger UNDEFINED operations.

1.3.4 Ranges And Extents

Ranges are specified by a pair of numbers separated by ".." and are inclusive; for example, a range of integers 0..4 includes the integers 0, 1, 2, 3, and 4.

Extents are specified by a pair of numbers in angle brackets separated by a colon and are inclusive, e.g., bits <7:3> specify an extent of bits including bits 7, 6, 5, 4, and 3.

1.3.5 Must be Zero (MBZ)

Fields specified as Must Be Zero (MBZ) must never be filled by software with a non-zero value. If the processor encounters a non-zero value in a field specified as MBZ, a Reserved Operand exception occurs.

1.3.6 Should be Zero (SBZ)

Fields specified as Should Be Zero (SBZ) should be filled by software with a zero value. These fields may be used at some future time. Non-zero values in SBZ fields produce UNPREDICTABLE results.

1.3.7 Read As Zero (RAZ)

Fields specified as Read As Zero (RAZ) return a zero when read.

1.3.8 Ignore (IGN)

Fields specified as Ignore (IGN) are ignored when written.

1.3.9 Register Format Notation

This specification contains a number of figures that show the format of various registers, followed by a description of each field. In general, the fields on the register are labeled with either a name or a mnemonic. The description of each field includes the name or mnemonic, the bit extent, and the type.

The "Type" column in the field description includes both the actual type of the field, and an optional initialized value, separated from the type by a comma. The type denotes the functional operation of the field, and may be one of the values shown in Table 1–1. If present, the initialized value indicates that the field is initialized by hardware to the specified value at power-up. If the initialized value is not present, the field is not initialized at power-up.

Table 1–1: Register Field Type Notation

Notation	Description
RW	A read-write bit or field. The value may be read and written by software.
RO	A read-only bit or field. The value may be read by software. It is written by hardware; software writes are ignored.
WO	A write-only bit or field. The value may be written by software. It is used by hardware and reads by software return an UNPREDICTABLE result.
WZ	A write bit or field. The value may be written by software. It is used by hardware and reads by software return a 0.
W1C	A write-one-to-clear bit. If reads are allowed to the register then the value may be read by software. If it is a write-only register then a read by software returns an UNPREDICTABLE result. Software writes of a 1 cause the bit to be cleared by hardware. Software writes of a 0 do not modify the state of the bit.
W0C	A write-zero-to-clear bit. If reads are allowed to the register then the value may be read by software. If it is a write-only register then a read by software returns an UNPREDICTABLE result. Software writes of a 0 cause the bit to be cleared by hardware. Software writes of a 1 do not modify the state of the bit.
WA	A write-anything-to-the-register-to-clear bit. If reads are allowed to the register then the value may be read by software. If it is a write-only register then a read by software returns an UNPREDICTABLE result. Software write of any value to the register cause the bit to be cleared by hardware.
RC	A read-to-clear field. The value is written by hardware and remains unchanged until read. The value may be read by software at which point, hardware may write a new value into the field.

In addition to named fields in registers, other bits of the register may be labeled with one of the three symbols listed in Table 1–2. These symbols denote the type of the unnamed fields in the register.

Table 1–2: Register Field Notation

Notation	Description
RAZ	Denotes a register bit(s) that is read as a 0.
IGN	Denotes a register bit(s) that is ignored on write and UNPREDICTABLE when read if not otherwise specified.

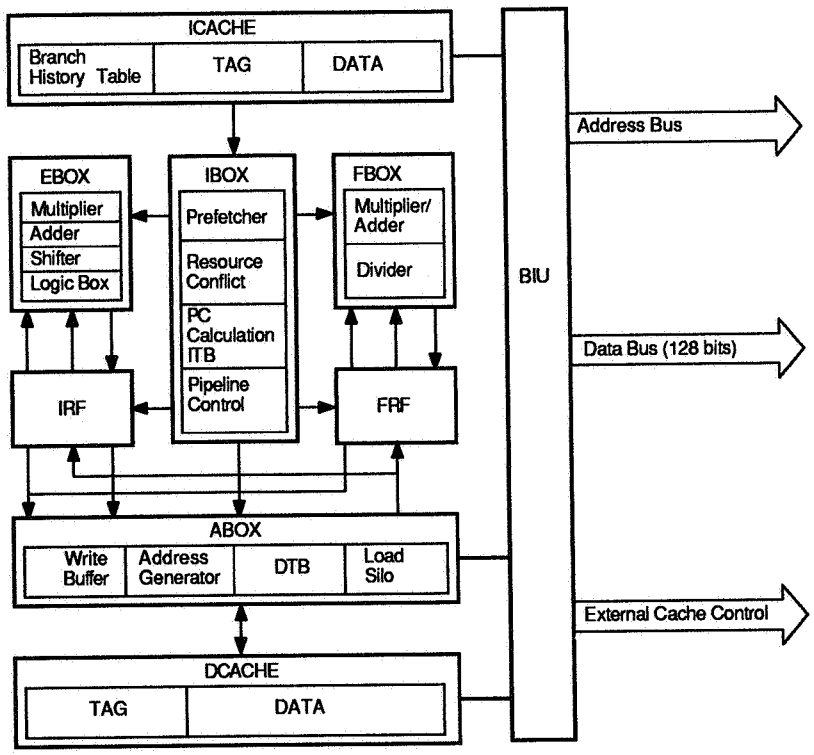
Chapter 2
Microarchitecture

2.1 Introduction

This chapter gives a view of the 21064 micro-architecture for programmers and system designers. It is not intended to be a detailed hardware description of the chip. This document first describes the hardware with only minimal forward references to the pipeline and then presents the pipeline. The 21064 CPU can issue two instructions in a single cycle - the scheduling and dual issue rules are defined at the end of the chapter.

It is important to realize that the Alpha architecture is the combination of the 21064 microarchitecture and PALcode. Many hardware design decisions were based on specific PAL functionality. These PAL assumptions and restrictions are detailed in the next chapter. It is important to keep in mind that if a certain piece of hardware appears to be "architecturally incomplete", the missing functionality is implemented in PALcode.

Figure 2–1: Block Diagram



2.2 Overview

The 21064 microprocessor consists of three independent execution units: integer execution unit (Ebox), floating point unit (Fbox), and the address generation, memory management, write buffer and bus interface unit (Abox). Each unit can accept at most one instruction per cycle, however if code is properly scheduled, the 21064 can issue two instructions to two independent units in a single cycle. A fourth unit, the Ibox, is the central control unit. It issues instructions, maintains the pipeline and performs all of the PC calculations. The 21064 also has on-chip instruction and data caches (Icache and Dcache).

2.3 Instruction Box - Ibox

The primary function of the Ibox is to issue instructions to the Ebox, Abox and Fbox. In order to provide those instructions, the Ibox also contains the prefetcher, PC pipeline, ITB, abort logic, register conflict or dirty logic, and exception logic. The Ibox decodes two instructions in parallel and checks that the required resources are available for both instructions. If resources are available and dual issue is possible then both instructions may be issued. The section on Dual Issue Rules details which instructions can be dual issued. If the resources are available for only the first instruction or the instructions cannot be dual issued then the Ibox issues only the first instruction. The Ibox does NOT issue instructions out of order, even if the resources are available for the second instruction and not for the first instruction. The Ibox does not issue instructions until the resources for the first instruction become available. If only the first of a pair of instructions issues, the Ibox does not advance another instruction to attempt to dual issue again. Dual issue is only attempted on aligned quadword pairs.

2.3.1 Branch Prediction Logic

The Ibox contains the branch prediction logic. The 21064 offers a choice of branch prediction strategies selectable through the ICCSR IPR. The Icache records the outcome of branch instructions in a single history bit provided for each instruction location in the cache. This information can be used as the prediction for the next execution of the branch instruction. The prediction for the first execution of a branch instruction is based on the sign of the displacement field within the branch instruction itself. If the sign bit is negative, conditional branches are predicted to be taken. If the sign is positive, conditional branches are predicted to be not taken. Alternatively, if the history table is disabled, branches can be predicted based on the sign of the displacement field at all times.

The 21064 provides a four entry subroutine return stack which is controlled by the hint bits in the BSR, HW_REI, and jump to subroutine instructions (JMP, JSR, RET, or JSR_COROUTINE). It also provide a means of disabling all branch prediction hardware.

2.3.2 Instruction Translation Buffer - ITB

The Ibox contains an 8-entry, fully associative translation buffer which caches recently used instruction-stream page table entries for 8Kbyte pages, and a four entry fully associative translation buffer which supports the largest granularity hint option (512*8Kbyte pages) as described in the Alpha Architecture Handbook. Both translation buffers use a not-last-used replacement algorithm. They are hereafter referred to as the small-page and large-page ITBs, respectively.

In addition, an extension is provided that is referred to as the super page, which can be enabled by the MAP bit in the ICCSR IPR. Super page mappings provide one-to-one virtual PC<33:13> to physical PC<33:13> translation when virtual address bits <42:41> = 2. This function essentially maps the entire physical address space multiple times over to one quadrant of the virtual address space defined by <42:41> = 2. When translating through the super page, the PTE[ASM] bit used in the Icache is always set. Access to the super page mapping is only allowed while executing in kernel mode.

The ITBs are filled and maintained by PALcode. The operation system via PALcode is responsible for insuring that virtual addresses can only be mapped through a single, large page, small page or super page ITB entry at the same time.

While not executing in PAL mode, the 43-bit virtual program counter (VPC) is presented each cycle to the ITB. If the PTE associated with the VPC is cached in the ITB then the PFN and protection bits for the page which contains the VPC are used by the Ibox to complete the address translation and access checks.

The 21064 ITB supports a single Address Space Number (ASN) via the PTE[ASM] bit. Each PTE entry in the ITB contains an ASM (address space match) bit. Writes to the ITBASM IPR invalidate all entries which do not have their ASM bit set. This provides a simple method of preserving entries which map operating system regions while invalidating all others.

2.3.3 Interrupt Logic

The 21064 chip supports three sources of interrupts; hardware, software and asynchronous system trap (AST). There are six level-sensitive hardware interrupts sourced by pins, 15 software interrupts sourced by an on-chip IPR (SIRR), and 4 AST interrupts sourced by a second internal IPR (ASTRR). All interrupts are independently maskable via on-chip enable registers to support a software controlled mechanism for prioritization. In addition, AST interrupts are qualified by the current processor mode and the current state of SIER[2].

By providing distinct enable bits for each independent interrupt source, a software controlled interrupt priority scheme can be implemented with maximum flexibility. For example, a six level interrupt priority scheme can be supported for the six hardware interrupt request pins by defining a distinct state of the corresponding hardware interrupt enable register for each IPL. The current interrupt priority is determined by the state of the interrupt enable register. The lowest interrupt priority level is produced by enabling all 6 interrupts, e.g bits 6-1. The next is produced by enabling bits 6-2 and so on to the highest interrupt priority level which is produced by enabling only bit 6 and disabling bits 5 through 1. When all interrupt enable bits are cleared, the processor can not be interrupted from the hardware interrupt request register. Each state, 6-1,6-2,6-3,6-4,6-5,6 represents an individual interrupt priority level (IPL). If these states are the only states allowed in the interrupt enable register, a six level hardware interrupt priority scheme can be controlled entirely by software.

The scheme is extendible to provide multiple interrupt sources at the same interrupt priority level by grouping enable bits. Groups of enable bits must be set and cleared together to support multiple interrupts of equal priority level. Of course, this method reduces the total available number of distinct levels.

Since enable bits are provided for all hardware, software and AST interrupt requests, a priority scheme can span all sources of processor interrupts. The only exception to this rule regards the restriction on AST interrupt requests as described below.

Four AST interrupts are provided; one for each processor mode. AST interrupt requests are qualified such that AST requests corresponding to a given mode are blocked whenever the processor is in a higher mode regardless of the state of the AST interrupt enable register. In addition, all AST interrupt requests are qualified in the 21064 with SIER[2].

When the processor receives an interrupt request and that request is enabled, an interrupt is reported or delivered to the exception logic if the processor is not currently executing PALcode. Before vectoring to the interrupt service PAL dispatch address, the pipeline is completely drained and all outstanding load instructions are completed. The restart address is saved in the Exception Address IPR (EXC_ADDR) and the processor enters PALmode. The cause

of the interrupt may be determined by examining the state of any of the interrupt request registers.

Note that hardware interrupt requests are level sensitive and therefore may be removed before an interrupt is serviced. If they are removed before the interrupt request register is read, the register will return a zero value.

2.3.4 Performance Counters

The 21064 chip contains a performance recording feature. The implementation of this feature provides a mechanism to count various hardware events and cause an interrupt upon counter overflow. Interrupts are triggered six cycles after the event, and therefore, the exception PC may not reflect the exact instruction causing counter overflow. Two counters are provided to allow accurate comparison of two variables under a potentially non-repeatable experimental condition. Counter inputs include issues, non-issues, total cycles, pipe dry, pipe freeze, mispredicts and cache misses as well as counts for various instruction classifications. In addition, one chip pin input to each counter is provided to measure external events at a rate determined by the selected system clock speed.

2.4 Exectution Box - Ebox

The Ebox contains the 64-bit integer execution datapath: adder, logic box, barrel shifter, byte zapper, bypassers and integer multiplier. The integer multiplier retires four bits per cycle. The Ebox also contains the 32-entry by 64-bit integer register file. This register file has four read ports and two write ports which allow the sourcing (sinking) of operands (results) to both the integer execution datapath and the Abox.

2.5 Address Box - Abox

The Abox contains six major sections: address translation datapath, load silo, write buffer, Dcache interface, IPRs and the external Bus Interface Unit (BIU). The address translation datapath has a displacement adder which generates the effective virtual address for load and store instructions, and a pair of translation buffers which generate the corresponding physical address.

2.5.1 Data Translation Buffer - DTB

The 21064 contains a 32-entry, fully associative translation buffer which caches recently used data-stream page table entries.and supports all four variants of the granularity hint option as described in the Alpha Architecture Handbook.

The 21064 provides an extension referred to as the super page, which can be enabled via ABOX_CTL<5:4>. Super page mappings provide virtual to physical address translation for two regions of the virtual address space. The first enables super page mapping when virtual address bits <42:41> = 2. In this mode, the entire physical address space is mapped multiple times over to one quadrant of the virtual address space defined by VA<42:41> = 2. The second super page mode maps a 30-bit region of the total physical address space defined by PA<33:30> = 0 into a single corresponding region of virtual space defined by VA<42:30> = 1FFE Hex. Super page translation is only allowed in kernel mode.

The operating system, via PALcode, is responsible for insuring that translation buffer entries, including super page regions, do not map overlapping virtual address regions at the same time.

The 21064 ITB supports a single Address Space Number (ASN) via the PTE[ASM] bit. Each PTE entry in the ITB contains an ASM (address space match) bit. Writes to the ITBASM IPR invalidate all entries which do not have their ASM bit set. This provides a simple method of preserving entries which map operating system regions while invalidating all others.

For load and store instructions, the effective 43-bit virtual address is presented to the DTBs. If the PTE of the supplied virtual address is cached in either DTB, the PFN and protection bits for the page which contains the address are used by the Abox to complete the address translation and access checks.

The DTBs are filled and maintained by PALcode. The chapter on PALcode details the DTB miss flow. Note that the DTBs can be filled in kernel mode by setting the HWE bit in the ICCSR IPR.

2.5.2 Bus Interface Unit - BIU

The BIU controls the interface to the 21064 pin bus. It responds to three classes of CPU generated requests: Dcache fills, Icache fills and write buffer-sourced commands. The BIU resolves simultaneous internal requests using a fixed priority scheme in which Dcache fill requests are given highest priority, followed by Icache fill requests. Write buffer requests have the lowest priority. The external interface chapter of this specification describes the 21064 pin bus. The BIU contains logic to directly access an external cache to service internal cache fill requests and writes from the write buffer. The BIU services reads and writes which do not hit in the external cache with help from external logic.

Internal data transfers between the CPU and the BIU are made via a 64-bit bidirectional bus. Since the internal cache fill block size is 32 bytes, cache fill operations result in four data transfers across this bus from the BIU to the appropriate cache. Also, since each write buffer entry is 32 bytes wide, write transactions may result in four data transfers from the write buffer to the BIU.

2.5.3 Load Silos

The Abox contains a fully folded memory reference pipeline which may accept a new load or store instruction every cycle until a Dcache fill is required. Since the Dcache lines are only allocated on load misses, the Abox may accept a new instruction every cycle until a load miss occurs. When a load miss occurs the Ibox stops issuing all instructions that use the load port of the register file or are otherwise handled by the Abox. This includes LDx, STx, HW_MTPR, HW_MFPR, FETCH, FETCH_M, RPCC, RS, RC, MB, and all memory format branch instructions, JMP, JSR, JSR_COROUTINE, and RET.

A JSR with a destination of R31 may be issued.

Since the result of each Dcache lookup is known late in the pipeline (stage [6]) and instructions are issued in pipe stage [3], there may be two instructions in the Abox pipeline behind a load instruction which misses the Dcache. These two instructions are handled as follows:

- Loads which hit the Dcache are allowed to complete - hit under miss.

- Load misses are placed in a silo and replayed in order after the first load miss completes.
- Store instructions are presented to the Dcache at their normal time with respect to the pipeline. They are siloed and presented to the write buffer in order with respect to load misses.

In order to improve performance, the Ibox is allowed to restart the execution of Abox directed instructions before the last pending Dcache fill is complete. Dcache fill transactions result in four data transfers from the BIU to the Dcache. These transfers may each be separated by one or more cycles depending on the characteristics of the external cache and memory subsystems. The BIU attempts to send the quadword of the fill block which the CPU originally requested in the first of these four transfers (it is always able to accomplish this for reads which hit in the external cache). Therefore the pending load instruction which requested the Dcache fill can complete before the Dcache fill finishes. Dcache fill data is not written into the cache array as it is received from the BIU. Rather, it accumulates one quadword at a time into a "pending fill" latch. When the load miss silo is empty and the requested quadword for the last outstanding load miss is received, the Ibox resumes execution of Abox directed instructions despite the still-pending Dcache fill. When the entire cache line has been received from the BIU, it is written into the Dcache data array whenever the array isn't otherwise busy with a load or a store.

2.5.4 Write Buffer

The Abox contains a write buffer for two purposes:

1. To minimize the number of CPU stall cycles by providing a high bandwidth (but finite) resource for receiving store data. This is required since the 21064 can generate store data at the peak rate of one quadword every CPU cycle which is greater than the rate at which the external cache subsystem can accept the data.
2. To attempt to aggregate store data into aligned 32-byte cache blocks for the purpose of maximizing the rate at which data may be written from the 21064 into the external cache.

In addition to store instructions, MB, STQ/C, STL/C, FETCH and FETCH_M instructions are also written into the write buffer and sent off-chip. Unlike stores, however, these write buffer-directed instructions are never merged into a write buffer entry with other instructions.

Each write buffer entry contains a CAM for holding physical address bits <33:5>, four quadwords of data, eight longword mask bits which indicate which of the associated eight longwords in the entry contain valid data, and miscellaneous control bits.

To facilitate the discussion, the following two states are defined: invalid and valid. A write buffer entry is invalid if it does not contain one of the above-listed write buffer-directed commands. A write buffer entry is valid if it contains one of the above-listed write buffer-directed commands.

The write buffer contains two pointers: the head pointer and the tail pointer. The head pointer points to the valid write buffer entry which has been valid the longest period of time. The tail pointer points to the invalid write buffer entry slot which will next be validated. If

the write buffer is completely full (empty) the head and tail pointers point to the same valid (invalid) entry.

Each time the write buffer is presented with a store instruction the physical address generated by the instruction is compared to the address in each valid write buffer entry. If the address is in the same aligned 32-byte block as an address in a valid write buffer entry which also contains a store then the new store data is merged into that entry, and the entry's longword mask bits are updated. If no matching address is found in the write buffer, then the store data is written into the entry designated by the tail pointer, the entry is validated, and the tail pointer is incremented to the next entry. Note this scheme does not maintain write-ordering.

The write buffer contains four entries and employs a complicated flow control mechanism which allows its entries to be efficiently used. The Ibox issues store instructions irrespective of whether the write buffer is full. If a store instruction enters pipe stage [6] of the Abox and the write buffer is full, the Ibox is forced to stop issuing both loads and stores by the same mechanism which is used for handling load misses. In effect, the store instruction gets treated as if it were a load miss. Any valid instructions in pipe stages [4] or [5] get handled exactly as if they had followed a load miss - loads which hit the Dcache are allowed to complete, stores are presented to the Dcache, placed into the Abox silo and and presented to the write buffer in order with respect to other siloed instructions. The Abox silo control logic insures that no stores are lost when the write buffer is full by retrying silo'ed stores until they are accepted by the write buffer.

The write buffer attempts to send its head entry off-chip by requesting the BIU when one of the following conditions are met:

1. The write buffer contains at least two valid entries.
2. The write buffer contains one valid entry and at least 256 CPU cycles have elapsed since the execution of the last write buffer-directed instruction.
3. The write buffer contains an MB instruction.
4. The write buffer contains a STQ/C or STL/C instruction.
5. A load miss is pending which requires the write buffer to be flushed before an external read is launched to service the load miss.

When the write buffer is requesting the BIU no stores are allowed to merge into the write buffer's head entry.

2.6 Fbox

The 21064 has an on-chip pipelined Fbox capable of executing both DEC and IEEE floating point instructions. IEEE floating point datatypes S and T are supported with all rounding modes except round to +/- infinity which can be provided in PALcode. DEC floating point datatypes F and G are fully supported with limited support for D floating format. The Fbox contains a 32-entry by 64-bit floating point register file and a user accessible control register, FPCR. The FPCR contains round mode controls, trap enables, and exception flag information. The Fbox can accept an instruction every cycle, with the exception of floating point divide instructions. The latency for data dependent, non divide instructions is six cycles. Bypass mechanisms are provided to allow issue of instructions which are dependent on prior results

while those results are written to the register file. For detailed information on instruction timing, refer to Section 2.9.

For divide instructions, the Fbox does not compute the inexact flag. Consequently, the INE exception flag in the FPCR register is never updated for any DIV instructions. This is a known incompatibility.

2.7 Cache Organization

The 21064 includes two on-chip caches, a data cache (Dcache) and an instruction cache (Icache). All memory cells in both Icache and Dcache are fully static, six transistor CMOS structures.

2.7.1 Data Cache

The 21064 Dcache contains 8K-bytes. It is a write-through, direct mapped, read-allocate physical cache and has 32-byte blocks. System components may keep the Dcache coherent with memory by using the invalidate bus described in the pin bus section of this specification.

2.7.2 Instruction Cache

The 21064 Icache is an 8Kbyte physical direct-mapped cache. Icache blocks, or lines, contain 32-bytes of instruction stream data with associated tag as well as a six-bit ASN field, a one-bit ASM field and an eight-bit branch history field per block. It does not contain hardware for maintaining coherency with memory and is unaffected by the invalidate bus.

The chip also contains a single-entry Icache stream buffer which together with its supporting logic reduces the performance penalty due to Icache misses incurred during in-line instruction processing. The stream buffer physically consists of latches for one Icache block's data and tag bits which are adjacent to the fill-side of the cache array, and a comparator, 13-bit incrementer and associated datapath hardware and control in the Abox.

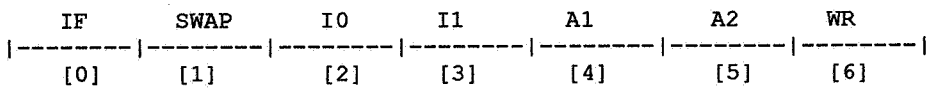
When an Icache miss occurs, the Ibox sends an Icache fill request to the Abox, which simultaneously requests the BIU and checks the stream buffer for the requested block. If the block is present in the stream buffer the Abox aborts the original Icache fill request, writes the requested block into the Icache and launches a prefetch request to the BIU for the next consecutive Icache block. The Ibox does not interact with the stream buffer; from the Ibox's perspective Icache misses which hit the stream buffer are the same as any other Icache miss except that the Icache fill finishes sooner.

When an Icache miss also misses the stream buffer the Abox launches a request for the required fill block and subsequently launches a prefetch request for the next consecutive fill block, thus getting the stream buffer started down the next I-stream path. Stream buffer prefetch requests never cross physical page boundaries, but instead wrap around to first block of the current page.

2.8 Pipeline Organization

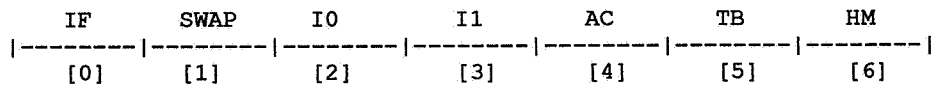
The 21064 has a seven stage pipeline for integer operate and memory reference instructions. Floating point operate instructions progress through a ten stage pipeline. The Ibox maintains state for all pipeline stages to track outstanding register writes, and determine Icache hit /miss. The pipeline diagrams below show the Ebox, Ibox, Abox and Fbox pipelines. The first four cycles are executed in the Ibox and the last stages are box specific. There are bypassers in all of the boxes that allow the results of one instruction to be used as operands of a following instruction without having to be written to the register file. The following section describes the pipeline scheduling rules.

Figure 2–2: Integer Operate Pipeline



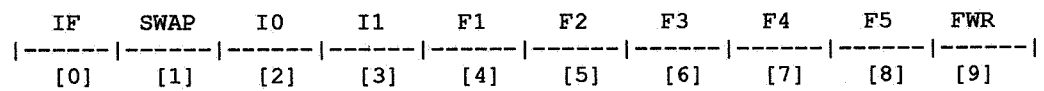
- IF - Instruction Fetch.
- SWAP - Swap Dual Issue Instruction /Branch Prediction.
- I0 - Decode.
- I1 - Register file(s) access / Issue check.
- A1 - Computation cycle 1 / Ibox computes new PC.
- A2 - Computation cycle 2 / ITB look-up
- WR - Integer register file write / Icache Hit/Miss

Figure 2–3: Memory Reference Pipeline



- AC - Abox calculates the effective D-stream address.
- TB - DTB look-up.
- HM - Dcache Hit/Miss and load data register file write

Figure 2–4: Floating Point Operate Pipeline



- F1-F5 - Floating point calculate pipeline

- FWR - Floating point register file write

The 21064 integer pipeline divides instruction processing into four static and three dynamic stages of execution. The 21064 floating point pipeline maintains the first four static stages and adds six dynamic stages of execution. The first four stages consist of the instruction fetch, swap, decode and issue logic. These stages are static in that instructions may remain valid in the same pipeline stage for multiple cycles while waiting for a resource or stalling for other reasons. Dynamic stages always advance state and are unaffected by any stall in the pipeline. Pipeline stalls are also referred to as pipeline freezes. A pipeline freeze may occur while zero instructions issue, or while one instruction of a pair issues and the second is held at the issue stage. A pipeline freeze implies that a valid instruction or instructions is (are) presented to be issued but can not proceed.

Upon satisfying all issue requirements, instructions are allowed to continue through any pipeline toward completion. After issuing, instructions cannot be held in a given pipe stage. It is up to the issue stage to insure that all resource conflicts are resolved before an instruction is allowed to continue. The only means of stopping instructions after the issue stage is an abort condition. Note that the term abort as used here is different from its use in the Alpha Architecture Handbook.

Aborts may result from a number of causes. In general, they may be grouped into two classes, namely exceptions (including interrupts) and non exceptions. The basic difference between the two is that exceptions require that the pipeline be drained of all outstanding instructions before restarting the pipeline at a redirected address. In either case, the pipeline must be flushed of all instructions which were fetched subsequent to the instruction which caused the abort condition. This includes stopping one instruction of a dual issued pair in the case of an abort condition on the first instruction of the pair. The non exception case, however, does not need to drain the pipeline of all outstanding instructions ahead of the aborting instruction. The pipeline can be immediately restarted at a redirected address. Examples of non exception abort conditions are branch mispredictions, subroutine call/return mispredictions and instruction cache misses. Data cache misses do not produce abort conditions but can cause pipeline freezes.

In the event of an exception, the processor aborts all instructions issued after the excepting instruction as described above. Due to the nature of some error conditions, this may occur as late as the write cycle. Next, the address of the excepting instruction is latched in the EXC_ ADDR IPR. When the pipeline is fully drained, the processor begins instruction execution at the address given by the PALcode dispatch. The pipeline is drained when all outstanding writes to both the integer and floating point register file have completed and all outstanding instructions have passed the point in the pipeline such that all instructions are guaranteed to complete without an exception in the absence of a machine check.

It should be noted that there are two basic reasons for non-issue conditions. The first is a pipeline freeze wherein a valid instruction or pair of instructions are prepared to issue but cannot due to a resource conflict. These type of non-issue cycles can be minimized through code scheduling. The second type of non-issue conditions consist of pipeline bubbles where there is no valid instruction in the pipeline to issue. Pipeline bubbles exist due to abort conditions as described above. In addition, a single pipeline bubble is produced whenever a branch type instruction is predicted to be taken, including subroutine calls and returns. Pipeline bubbles are reduced directly by the hardware through bubble squashing, but can also be effectively minimized through careful coding practices. Bubble squashing involves

the ability of the first four pipeline stages to advance whenever a bubble is detected in the pipeline stage immediately ahead of it while the pipeline is otherwise frozen.

2.9 Scheduling and Issuing Rules

2.9.1 Instruction Class Definition

It is important to note that the following scheduling and dual issue rules are only performance related. There are no functional dependencies related to scheduling or dual issuing. The scheduling and issuing rules are defined in terms of instruction classes. The table below specifies all of the instruction classes and the box which executes the particular class.

Table 2–1: Producer-Consumer Classes

Class Name	Box	Instruction List
LD	Abox	all loads, (HW_MFPR, RPCC, RS, RC, STC producers only), (FETCH consumer only)
ST	Abox	all stores, HW_MTPR
IBR	Ebox	integer conditional branches
FBR	Fbox	floating point conditional branches
JSR	Ebox	jump to subroutine instructions JMP, JSR, RET, or JSR_COROUTINE, (BSR, BR producer only)
IADDLOG	Ebox	ADDL ADDL/V ADDQ ADDQ/V SUBL SUBL /V SUBQ SUBQ/V S4ADDL S4ADDQ S8ADDL S8ADDQ S4SUBL S4SUBQ S8SUBL S8SUBQ LDA LDAH AND BIS XOR BIC ORNOT EQV
SHIFTCM	Ebox	SLL SRL SRA EXTQL EXTLL EXTWL EXTBL EXTQH EXTLH EXTWH MSKQL MSKLL MSKWL MSKBL MSKQH MSKLH MSKWH INSQL INSLL INSWL INSBL INSQH INSLH INSWH ZAP ZAPNOT CMOVEQ CMOVNE CMOVLT CMOVLE CMOVGT CMOVGE CMOVLBS CMOVLBC
ICMP	Ebox	CMPEQ CMPLT CMPLE CMPULT CMPULE CMPBGE
IMULL	Ebox	MULL MULL/V
IMULQ	Ebox	MULQ MULQ/V UMULH
FPOP	Fbox	floating point operates except divide
FDIV	Fbox	floating point divide

2.9.2 Producer-Consumer Latency Matrix

The 21064 enforces the following issue rules regarding producer/consumer latencies.

The scheduling rules are described as a producer-consumer matrix. Each row and column in the matrix is a class of Alpha instructions. A '1' in the Producer-Consumer Latency Matrix indicates one cycle of latency. A one cycle latency means that if instruction B uses the results of instruction A, then instruction B may be issued ONE cycle after instruction A is issued.

The first thing to do when determining latency for a given instruction sequence is to identify the classes of all the instructions. The example below has the classes listed in the comment field.

```
ADDQ    R1, R2, R3      ! IADDLOG class
SRA     R3, R4, R5      ! SHIFT class
SUBQ    R5, R6, R7      ! IADDLOG class
STQ     R7, D(R10)      ! ST class
```

The SRA instruction consumes the result (R3) produced by the ADDQ instruction. The latency associated with an iadd-shift producer-consumer pair as specified by the matrix is one. That means that if the ADDQ was issued in cycle 'n' the SRA could be issued in cycle 'n+1'. The SUBQ instruction consumes the result (R5) produced by the SRA instruction. The latency associated with a shift-iadd producer-consumer pair as specified by the matrix is two. That means that if the SRA was issued in cycle 'n' the SUBQ could be issued in cycle 'n+2'. The Ibox injects one nop cycle in the pipeline for this case.

The final case has the STQ instruction consuming the result (R7) produced by the SUBQ instruction. The latency associated with an iadd-st producer-consumer pair where the result of the iadd is the store data is zero. This means that the SUBQ and STQ instruction pair can be dual-issued.

Figure 2–5: Producer-Consumer Latency Matrix

Producer Class

Figure 2–5 (continued on next page)

Figure 2–5 (Cont.): Producer-Consumer Latency Matrix

Consumer Class	LD (1)	JSR	IADDLOG	SHIFTCM	ICMP	IMUL	FBR	FPOP	FDIV
LD	3	3	2	2	2	21	23	X	X	X	X
ST (2)	3	3	2/0	2/0	2/0	21/20	23/22	X/4	X/32	X/61
IBR	3	3	1	2	1	21	23	X	X	X
JSR	3	3	2	2	2	"	"	X	X	X
IADDLOG	3	3	1	2	2	"	"	X	X	X
SHIFTCM	3	3	1	2	2	"	"	X	X	X
ICMP	3	3	1	2	2	"	"	X	X	X
IMUL	3	3	1	2	2	21/19	23/21	X	X	X
FBR	3	X	X	X	X	X	X	6	34	63
FPOP	3	X	X	X	X	X	X	6	34	63
FDIV	3	X	X	X	X	X	X	6	34/30	63/59

Notes:

1. For loads, Dcache hit is assumed. The latency for a Dcache miss and an external cache hit is dependent on the system configuration. The latency is determined as the register file write time less 1 cycle.
2. For some producer classes, two latencies, X/Y, are given with the ST consumer class. X represents the latency for base address of store and Y represents the latency for store data. FDIV results cannot be used as the base address for store operations.
3. For IMUL followed by IMUL, there are two latencies given. The first represents the latency with data dependency, i.e. the second IMUL uses the result from the first. The second is the multiply latency without data dependencies.
4. For FDIV followed by FDIV, there are two latencies given. The first represents the latency with data dependency, i.e. the second FDIV uses the result from the first. The second is the division latency without data dependencies.

2.9.3 Producer-Producer Latency

Producer-producer latency, also known as write after write conflicts, are restricted only by the register write order. For most instructions, this is dictated by issue order, however IMUL, FDIV and LD instructions may require more time than other instructions to complete and therefore must stall following instructions that write the same destination register to preserve write ordering. In general, only cases involving an intervening producer-consumer conflict are of interest. They can occur commonly in a dual issue situation when a register is reused. In these cases, producer-consumer latencies are equal to or greater than the required producer-producer latency as determined by write ordering and therefore dictate the overall latency. An example of this case is shown in the code:

```
LDQ R2,D(R0)   ; R2 destination
ADDQ R2,R3,R4 ; wr-rd conflict stalls execution waiting for R2
LDQ R2,D(R1)   ; wr-wr conflict may dual issue when addq issues
```

2.9.4 Instruction Issue Rules

The following is a list of conditions that prevent instruction issue:

1. No instruction can be issued until all of it's source and destination registers are clean, i.e. all outstanding writes to the destination register are guaranteed to complete in issue order and there are no outstanding writes to the source registers or those writes can be bypassed.
2. No LD, ST, FETCH, MB, RPCC, RS, RC, TRAPB, HW_MXPR or BSR,BR,JSR(with destination other than R31) can be issued after a MB instruction until the MB has been acknowledged on the external pin bus.
3. No IMUL instructions can be issued if the integer multiplier is busy.
4. No SHIFT, IADDLOG, ICMP or ICMOV instruction can be issued exactly three cycles before an integer multiplication completes.
5. No integer or floating point conditional branch instruction can be issued in the cycle immediately following a JSR,JMP,RET,JSR_COROUTINE or HW_REI instruction.
6. No TRAPB instruction can be issued as the second instruction of a dual issue pair.
7. No LD instructions can be issued in the two cycles immediately following a STC.
8. No LD, ST, FETCH, MB, RPCC, RS, RC, TRAPB, HW_MXPR or BSR,BR,JSR(with destination other than R31) instruction can be issued when the Abox is busy due to a load miss or write buffer overflow. For more information see section 2.5.3.
9. No FDIV instruction can be issued if the floating pointer divider is busy.
10. No floating point operate instruction can be issued exactly five or exactly six cycles before the floating point divide completes.

2.9.5 Dual Issue Rules

The table below lists the classes of instruction pairs that can be issued in a single cycle. An instruction from a class in the first column below may be issued in the same cycle as an instruction from a class in the second column, in the absence of data dependencies and if the two instructions occupy the same aligned quadword in memory.

Table 2–2: Dual Issue Rules

Instruction 1	Instruction 2
LD integer	LD floating pt
LD floating pt	LD integer
ST floating pt	ST integer
FBR	IBR
IADDLOG	FPOP
SHIFT	FDIV
ICMP	JSR
ICMOV	BSR
IMUL	BR
	HW_x
	CALL_PAL

- No more than one of LD, ST, HW_MxPR, FETCH, RPCC, RS, RC, MB, TRAPB, BSR, BR or JSR can be issued in the same cycle.
- No more than one of JSR, IBR, BSR, HW_REI, BR or FBR can be issued in the same cycle.

Chapter 3
Privileged Architecture Library Code

3.1 Introduction

In a family of machines both users and operating system implementers require functions to be implemented consistently. When functions are implemented to a common interface, the code that uses those functions can be used on several different implementations without modification.

These functions range from the binary encoding of the instructions and data, to the exception mechanisms and synchronization primitives. Some of these functions can be cost effectively implemented in hardware. However several of these functions are impractical to implement directly in hardware. They include low-level hardware support functions such as translation buffer fill routines, interrupt acknowledge, and exception dispatch. Also included is support for privileged and atomic operations that require long instruction sequences such as Return from Exception or Interrupt (REI).

In a CISC architecture, these functions are generally provided by the use of microcode. In the 21064 RISC microprocessor, there is no microcode. Instead there is an architected interface to functions that will be consistent with other members of the Alpha family of machines. The Privileged Architecture Library Code (PALcode) is used to implement these functions without resorting to a microcoded machine.

3.2 PAL Environment

PALcode runs in an environment with privileges enabled, instruction stream mapping disabled, and interrupts disabled. The enabling of privileges allows all functions of the machine to be controlled. Disabling of instruction stream mapping allows PALcode to be used to support the memory management functions (e.g., translation buffer fill routines can not be run via mapped memory).

PALcode can perform both virtual and physical data stream references. The disabling of interrupts allows the system to provide multi-instruction sequences as atomic operations. The PALcode environment in the 21064 also includes 32 PAL temp registers which are accessible only by special PAL instructions.

3.3 Special PAL Instructions

PALcode uses the Alpha instruction set for most of its operations. The 21064 maps the architecturally reserved PALcode opcodes (PALRES0 - PALRES4) to a special load and store (HW_LD, HW_ST), a move to and move from processor register (HW_MTPR, HW_MFPR), and a return from PALmode exception (HW_REI). These instructions produce a Reserved Opcode fault if executed while not in the PALcode environment unless the HWE bit of the ICCSR IPR is set, in which case these instructions can be executed in kernel mode.

Register checking and bypassing logic is provided for PALcode instructions as it is for non-PALcode instructions when using general purpose registers. Explicit software timing is required for accessing the hardware specific IPRs and the PAL_TEMPs. These constraints are described in the PALmode restriction and IPR sections.

3.3.1 HW_MFPR and HW_MTPR

The internal processor register specified by the PAL, ABX, IBX, and index field is written /read with the data from the specified integer register. Processor registers may have side effects that happen as the result of writing/reading them. Coding restrictions are associated with accessing various registers. Separate bits are used to access Abox IPRs, Ibox IPRs, and PAL_TEMPs, therefore it is possible for an MTPR instructions to write multiple registers in parallel if they both have the same index.

The HW_MFPR and HW_MTPR instructions have the following format:

Figure 3–1: HW_MxPR Instruction Format

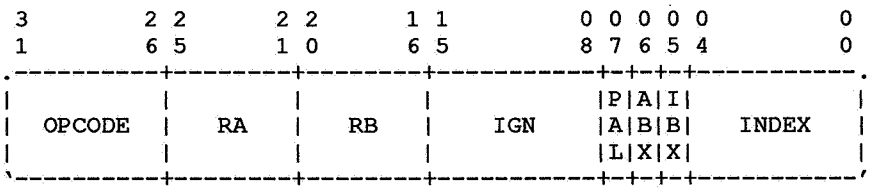


Table 3–1: HW_MFPR and HW_MTPR Format Description

Field	Description
OPCODE	Is 25 (HW_MFPR) or 29 (HW_MTPR), decimal.
RA/RB	Contain the source,HW_MTPR or destination,HW_MFPR, register number. The RA and RB fields must always be identical.
PAL	If set, this HW_MFPR or HW_MTPR instruction is referencing a PAL temporary register, PAL_TEMP.
ABX	If set, this HW_MFPR or HW_MTPR instruction is referencing a register in the Abox.
IBX	If set, this HW_MFPR or HW_MTPR instruction is referencing a register in the Ibox.

Table 3–1 (Cont.): HW_MFPR and HW_MTPR Format Description

Field	Description
INDEX	Specifies hardware specific register as shown in Table 3–2

The following table indicates how the PAL, ABX, IBX, and INDEX fields are set to access the internal processor registers. Setting the PAL, ABX, and IBX fields to zero generates a NOP.

Table 3–2: IPR Access

Mnemonic	PAL	ABX	IBX	INDEX	Access	Comments
TB_TAG	x	x	1	0	W	PAL mode only
ITB_PTE	x	x	1	1	R/W	PAL mode only
ICCSR	x	x	1	2	R/W	
ITB_PTE_TEMP	x	x	1	3	R	PAL mode only
EXC_ADDR	x	x	1	4	R/W	
SL_RCV	x	x	1	5	R	
ITBZAP	x	x	1	6	W	PAL mode only
ITBASM	x	x	1	7	W	PAL mode only
ITBIS	x	x	1	8	W	PAL mode only
PS	x	x	1	9	R/W	
EXC_SUM	x	x	1	10	R/W	
PAL_BASE	x	x	1	11	R/W	
HIRR	x	x	1	12	R	
SIRR	x	x	1	13	R/W	
ASTRR	x	x	1	14	R/W	
HIER	x	x	1	16	R/W	
SIER	x	x	1	17	R/W	
ASTER	x	x	1	18	R/W	
SL_CLR	x	x	1	19	W	
SL_XMIT	x	x	1	22	W	
TB_CTL	x	1	x	0	W	
DTB_PTE	x	1	x	2	R/W	
DTB_PTE_TEMP	x	1	x	3	R	

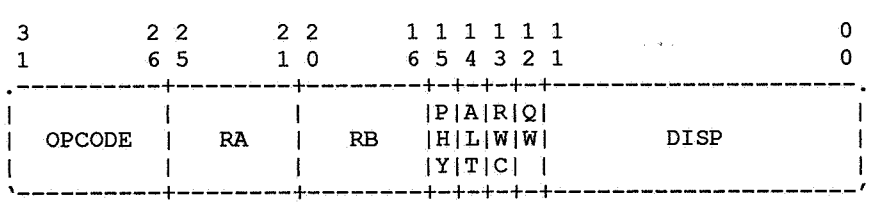
Table 3–2 (Cont.): IPR Access

Mnemonic	PAL	ABX	IBX	INDEX	Access	Comments
MMCSR	x	1	x	4	R	
VA	x	1	x	5	R	
DTBZAP	x	1	x	6	W	
DTASM	x	1	x	7	W	
DTBIS	x	1	x	8	W	
BIU_ADDR	x	1	x	9	R	
BIU_STAT	x	1	x	10	R	
DC_ADDR	x	1	x	11	R	
DC_STAT	x	1	x	12	R	
FILL_ADDR	x	1	x	13	R	
ABOX_CTL	x	1	x	14	W	
ALT_MODE	x	1	x	15	W	
CC	x	1	x	16	W	
CC_CTL	x	1	x	17	W	
BIU_CTL	x	1	x	18	W	
FILL_SYNDROME	x	1	x	19	R	
BC_TAG	x	1	x	20	R	
FLUSH_IC	x	1	x	21	W	
FLUSH_IC_ASM	x	1	x	23	W	
PAL_TEMP[31..0]	1	x	x	31-00	R/W	

3.3.2 HW_LD and HW_ST

PALcode uses the HW_LD and HW_ST instructions to access memory outside of the realm of normal Alpha memory management. The HW_LD and HW_ST instructions have the following format:

Figure 3–2: HW_LD/HW_ST Instruction Format



The effective address of these instructions is calculated as follows:

addr <- (SEXT(DISP) + RB) AND NOT (QW | 11(bin))

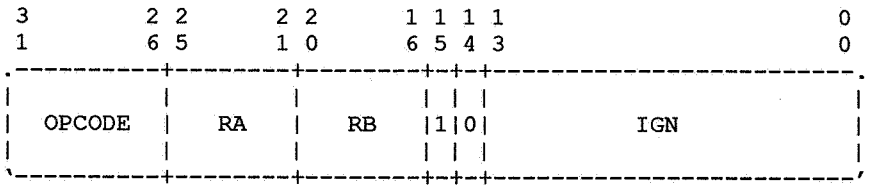
Table 3–3: HW_LD and HW_ST Format Description

Field	Description
OPCODE	Is 27 (HW_LD) or 31 (HW_ST), decimal.
RA/RB	Contain register numbers, interpreted in the normal fashion for loads and stores.
PHY	If clear, the effective address of the HW_LD or HW_ST is a virtual address. If set then the effective address of the HW_LD or HW_ST is a physical address.
ALT	For virtual-mode HW_LD and HW_ST instructions this bit selects the processor mode bits which are used for memory management checks. If ALT is clear the current mode bits of the PS register are used, while if ALT is set the mode bits in the ALT_MODE IPR are used. Physical-mode load-lock and store-conditional variants of the HW_LD and HW_ST instructions may be created by setting both the PHY and ALT bits.
RWC	The RWC (read with write check) bit, if set, enables both read and write access checks on virtual HW_LD instructions.
QW	The quadword bit specifies the data length. If it is set then the length is quadword. If it is clear then the length is longword.
DISP	The DISP field holds a 12-bit signed byte displacement.

3.3.3 HW_REI

The HW_REI instruction uses the address in the Ibox EXC_ADDR IPR to determine the new virtual program counter (VPC). Bit zero of the EXC_ADDR indicates the state of the PALmode bit on the completion of the HW_REI. If EXC_ADDR bit[0] is set then the processor remains in PALmode. This allows PALcode to transition from PALmode to non-PALmode. The HW_REI instruction can also be used to jump from PALmode to PALmode. This allows PAL instruction flows to take advantage of the D-stream mapping hardware in the 21064, including traps. The HW_REI instruction has the following format:

Figure 3–3: HW_REI Instruction Format



Note that bits[15..14] contain the branch prediction hint bits. The 21064 pushes the contents of the EXC_ADDR register on the JSR prediction stack. Bit[15] must be set to pop the stack to avoid misalignment. The next address and PALmode bit are calculated as follows:

```
VPC <- EXC_ADDR AND {NOT 3}
PALmode <- EXC_ADDR[0]
```

Table 3–4: The HW_REI Format Description

Field	Description
OPCODE	The OPCODE field contains 30, decimal.
RA/RB	Contain register numbers which should be R31 or a stall may occur.

3.4 PAL Entry Points

When an exception or interrupt occurs, the 21064 first drains the pipeline, loads the PC into the EXC_ADDR IPR and then dispatches to one of the exception routines. The pipeline is drained when all instructions that update either register file have completed, and all instructions that do not update the register files are guaranteed to complete without an exception in the absence of a machine check. In addition, all pending Dcache fill operations must have completed before dispatch to one of the exception routines. If multiple exceptions occur, the 21064 dispatches to the highest priority PAL entry point. The table below prioritizes entry points from highest to lowest priority, i.e. the first row in the table (reset) has the highest priority.

The table below defines only the entry point offset, bits [13..0]. The high-order bits of the new PC (bits [33..14]) come from the PAL_BASE IPR.

Note that PALcode at PAL entry points of higher priority than DTBMISS must unlock possible MMCSR IPR and VA IPR locks.

Table 3–5: PAL Entry Points

Entry Name	Time	Offset(Hex)	Cause
RESET	anytime	0000	

Table 3–5 (Cont.): PAL Entry Points

Entry Name	Time	Offset(Hex)	Cause
MCHK	pipe_stage[7]	0020	Uncorrected hardware error.
ARITH	anytime	0060	Arithmetic exception.
INTERRUPT	anytime	00E0	Includes corrected hardware error.
D-stream errors	pipe_stage[6]	01E0, 08E0, 09E0, 11E0	See Table 3–6.
ITB_MISS	pipe_stage[5]	03E0	ITB miss.
ITB_ACV	pipe_stage[5]	07E0	I-stream access violation.
CALL_PAL	pipe_stage[5]	2000,40,80,C0 thru 3FC0	128 locations based on instruction[7,5:0]. See description below.
OPCDEC	pipe_stage[5]	13E0	Reserved or privileged opcode.
FEN	pipe_stage[5]	17E0	FP op attempted with :
			FP instructions disabled via ICCSR FPE bit
			FP IEEE round to +/- infinity
			FP IEEE with datatype field other than S,T,QW

PALcode functions are implemented via the CALL_PAL instruction. CALL_PAL instructions cause exceptions in the hardware. As with all exceptions, the EXC_ADDR register is loaded by hardware with a possible return address.

The 21064 loads the EXC_ADDR register with the address of the instruction *following* the CALL_PAL. For this reason, PALcode supporting the desired PAL mode function should not increment the EXC_ADDR register before executing a HW_REI instruction to return to native mode. This feature requires special handling in the arithmetic trap and machine check PALcode flows. See Section 3.8.6 EXC_ADDR for more complete information.

To improve speed of execution, a limited number of CALL_PAL instructions are directly supported in hardware with dispatches to specific address offsets.

The 21064 provides the first 64 privileged and 64 unprivileged CALL_PAL instructions with code regions of 64 bytes. This produces hardware PAL entry points as described below.

Privileged CALL_PAL Instructions [00000000:0000003F]

Offset(Hex) = 2000 + (<5:0> shift left 6)

Unprivileged CALL_PAL Instructions [00000080:000000BF]

Offset(Hex) = 3000 + (<5:0> shift left 6)

The CALL_PAL instructions that do not fall within the ranges [00000000:0000003F] and [00000080:000000BF] result in an OPCDEC exception. In addition, CALL_PAL instructions that fall within the range [00000000:0000003F] while the 21064 is not executing in kernel mode will result in an OPCDEC exception.

The PAL entry points assigned to D-stream errors require a bit more explanation. The hardware recognizes four classes of D-stream memory management errors: bad virtual address (improper sign extension), DTB miss, alignment error and everything else (ACV, FOR, FOW). These errors get mapped into four PAL entry points: UNALIGN, DTB_MISS PAL mode, DTB_MISS Native mode and D_FAULT. Table 3–6 lists the priority of these entry points as a group with respect to each of the other entry points. Since a particular D-stream memory reference may generate errors which fall into more than one of the four error classes which the hardware recognizes, we also must define the priority of each of the D-stream PAL entry points with respect to the others in the D-stream PAL entry group. Table 3–6 gives this priority.

Table 3–6: D-stream Error PAL Entry Points

BAD_VA	DTB_ MISS	UNALIGN	PAL	Other	Offset(Hex)
1	x	0	x	x	01E0 D_FAULT
1	x	1	x	x	11E0 UNALIGN
0	1	x	0	x	08E0 DTB_MISS Native
0	1	x	1	x	09E0 DTB_MISS PAL
0	0	1	x	x	11E0 UNALIGN
0	0	0	x	1	01E0 D_FAULT

3.5 PALmode Restrictions

Many of the restrictions involve waiting 'n' cycles before using the results of PAL instructions. Inserting 'n' instructions between the two time-sensitive instructions is the typical method of waiting for 'n' cycles. Because the 21064 can dual issue instructions it is possible to write code that requires 2*n+1 instructions to wait 'n' cycles. Due to the resource requirements of individual instructions, and the 21064 hardware design, multiple copies of the same instruction can not be dual issued. This fact is used in some of the code examples below.

1. As a general rule, HW_MTPR instructions require at least 4 cycles to update the selected IPR. Therefore, at least three cycles of delay must be inserted before using the result of the register update.

   Note that only the write followed by read operation requires this software timing. Multiple reads, multiple writes, or read followed by write will pipeline properly and do not require software timing except for accesses of the TB registers.

   These cycles can be guaranteed by either including seven instructions which do not use the IPR in transition or proving through the dual issue rules and/or state of the machine, that at least three cycles of delay will occur. As a special case, multiple copies of a HW_MTPR instruction, used as a NOP instruction, can be used to pad cycles after the original HW_MTPR. Since multiple copies of the same instruction will never dual issue, the maximum number of instructions necessary to insure at least three cycles of delay is three.

An example of this is :

```
HW_MTPR Rx, HIER        ; Write to HIER
HW_MFPR R31, 0          ; NOP hw_mxpr instruction
HW_MFPR R31, 0          ; NOP hw_mxpr instruction
HW_MFPR R31, 0          ; NOP hw_mxpr instruction
HW_MFPR Ry, HIER        ; Read from HIER
```

The HW_REI instruction uses the ITB if the EXC_ADDR register contains a non PAL mode VPC, VPC<0> = 0. By the rule above, this implies that at least 3 cycles of delay must be included after writing the ITB before executing a HW_REI instruction to exit PAL mode.

Exceptions:

- HW_MFPR instructions reading any PAL_TEMP register can never occur exactly two cycles after a HW_MTPR instruction writing any PAL_TEMP register. The simple solution results in code of the form:

```
HW_MTPR Rx, PAL_R0      ; Write PAL temp 0
HW_MFPR R31, 0          ; NOP hw_mxpr instruction
HW_MFPR R31, 0          ; NOP hw_mxpr instruction
HW_MFPR R31, 0          ; NOP hw_mxpr instruction
HW_MFPR Ry, PAL_R0      ; Read PAL temp 0
```

  The above code guarantees three cycles of delay after the write before the read. It is also possible to make use of the cycle immediately following a HW_MTPR to execute a HW_MFPR instruction from a different PAL_TEMP register. This requires that the second instruction not stall for exactly one cycle. This is much easier to insure. A HW_MFPR instruction may stall for a single cycle as a result of a write after write conflict.

- The EXC_ADDR register may be read by a HW_REI instruction only two cycles after the HW_MTPR. This is equivalent to one intervening cycle of delay. This translates to code of the form:

```
HW_MTPR Rx, EXC_ADDR    ; Write EXC_ADDR
HW_MFPR R31, 0          ; NOP cannot dual issue with either
HW_REI                  ; Return
```

2. An MTPR operation to the DTBIS register cannot be bypassed into. In other words, all data being moved to the DTBIS register must be sourced directly from the register file. One way to insure this is to provide at least three cycles of delay before using the result of any integer operation (except MUL) as the source of an MTPR DTBIS. Do not use a MUL as the source of DTBIS data. Sample code for this operation is :

```
ADDQ R1,R2,R3           ; source for DTBIS address
ADDQ R31,R31,R31        ; cannot dual issue with above, 1st
                        ;  cycle of delay
ADDQ R31,R31,R31        ; 2nd cycle of delay
ADDQ R31,R31,R31        ; 3rd cycle of delay
ADDQ R31,R31,R31        ; may dual issue with below, else
                        ;  4th cycle of delay
HW_MTPR R3,DTBIS        ; R3 must be in register file, no
                        ;  bypass possible
```

3. At least one cycle of delay must occur after a HW_MTPR TB_CTL before either a HW_MTPR ITB_PTE or a HW_MFPR ITB_PTE to allow setup of the ITB large page/small page decode.
4. The first cycle (the first one or two instructions) at all PALcode entry points may not execute a conditional branch instruction or any other instruction that uses the jsr stack hardware. This includes instructions JSR,JMP,RET,COROUTINE, BSR,HW_REI and all Bxx opcodes except BR, which is allowed.
5. The following table indicates the number of cycles required after a HW_MTPR instruction before a subsequent HW_REI instruction for the specified IPRs. These cycles can be insured by inserting one HW_MFPR R31,0 instruction or other appropriate instruction(s) for each cycle of delay required after the HW_MTPR.

Table 3–7: HW_MTPR/HWMTPR Restrictions

IPR	Cycles between HW_MTPR and HW_REI
DTBIS,ASM,ZAP	0
ITBIS,ASM,ZAP	2
xIER	3
xIRR	3
PS	4
ICCSR<FPE>	3
ICCSR<ASN>	9 ; Cycles before HW_REI
FLUSH_IC[ASM]	9 ; target can be fetched.

6. When loading the CC register, bits <3:0> must be loaded with zero. Loading non-zero values in these bits may cause the count to be inaccurate.
7. An MTPR DTBIS cannot be combined with an MTPR ITBIS instruction. The hardware will not clear the ITB if both the Ibox and Abox IPRs are simultaneously selected. Instead, two instructions are needed to clear each TB individually. Code example:

```
HW_MTPR Rx,ITBIS
HW_MTPR Ry,DTBIS
```

8. Three cycles of delay are required between:

```
MTxIER and MFxIRR
MTxIRR and MFxIRR
MTPS   and LD or ST, all
MTPS   and MFxIRR
```

9. A HW_MxPR ITB_TAG, ITB_PTE, ITB_PTE_TEMP cannot follow a HW_REI that remains in PAL mode. (Address bit<0> of the EXC_ADDR is set) This rule implies that it is not a good idea to ever allow exceptions while updating the ITB. If an exception interrupts flow of the ITB fill routine and attempts to REI back, and the return address begins with a HW_MxPR instruction to an ITB register, and the REI is predicted correctly to avoid any delay between the two instructions, then the ITB register will not be written. Code example:

```
HW_REI               ; return from interrupt
HW_MTPR R1,ITB_TAG   ; attempts to execute very next
                     ;  cycle, instr ignored
```

10. The ITB_TAG,ITB_PTE and ITB_PTE_TEMP registers can only be accessed in PAL mode. If the instructions HW_MTPR or HW_MFPR to/from the above registers are attempted while not in PAL mode by setting the HWE (hardware enable) bit of the ICCSR, the instructions will be ignored.

11. Machine check exceptions taken while in PAL mode may load the EXC_ADDR register with a restart address one instruction earlier than the proper restart address. Some HW_MxPR instructions may have already completed execution even though the restart address indicates the HW_MxPR as the return instruction. Re-execution of some HW_MxPR instructions can alter machine state. (e.g. TB pointers, EXC_ADDR register mask)

    The mechanism used to stop instruction flow during machine check exceptions causes the machine check exception to appear as a D-stream fault on the following instruction in the hardware pipeline. In the event that the following instruction is a HW_MxPR, a D-stream fault will not abort execution in all cases. Although the EXC_ADDR will be loaded with the address of the HW_MxPR instruction as if it were aborted, a HW_REI to this restart address will incorrectly re-execute this instruction.

    Machine check service routines should check for HW_MxPR instructions at the return address before continuing.

12. When writing the PAL_BASE register, exceptions may not occur. An exception occurring simultaneously with a write to the PAL BASE may leave the register in a metastable state. All asynchronous exceptions but reset can be avoided under the following conditions:

```
PAL mode ..................... blocks all interrupts
machine checks disabled ..... blocks I/O error exceptions
  (via ABOX_CTL reg or MB isolation)
Not under trap shadow ....... avoids arithmetic traps
```

The trap shadow is defined as less than:

```
 3 cycles after a non-mul integer operate that may overflow
22 cycles after a MULL/V instruction
24 cycles after a MULQ/V instruction
 6 cycles after a non-div fp operation that may cause a trap
34 cycles after a DIVF or DIVS that may cause a trap
63 cycles after a DIVG or DIVT that may cause a trap
```

13. The sequence HW_MTPR PTE, HW_MTPR TAG is NOT allowed. At least two null cycles must occur between HW_MTPR PTE and HW_MTPR TAG.

14. The AMCHK exception service routine must check the EXC_SUM register for simultaneous arithmetic errors. Arithmetic traps will not trigger exceptions a second time after returning from exception service for the machine check.

15. Three cycles of delay must be inserted between HW_MFPR DTB_PTE and HW_MFPR DTB_PTE_TEMP. Code example:

```
HW_MFPR Rx,DTB_PTE      ; reads DTB_PTE into DTB_PTE_TEMP
                        ;  register
HW_MFPR R31,0           ; 1st cycle of delay
HW_MFPR R31,0           ; 2nd cycle of delay
HW_MFPR R31,0           ; 3rd cycle of delay
HW_MFPR Ry,DTB_PTE_TEMP ; read DTB_PTE_TEMP into register
                        ;  file Ry
```

16. Three cycles of delay must be inserted between HW_MFPR IPTE and HW_MFPR ITB_PTE_TEMP. Code example:

```
HW_MFPR Rx,DTB_PTE      ; reads DTB_PTE into DTB_PTE_TEMP
                        ;  register
HW_MFPR R31,0           ; 1st cycle of delay
HW_MFPR R31,0           ; 2nd cycle of delay
HW_MFPR R31,0           ; 3rd cycle of delay
HW_MFPR Ry,DTB_PTE_TEMP ; read DTB_PTE_TEMP into register
                        ;  file Ry
```

17. The content of the destination register for HW_MFPR Rx,DTB_PTE or HW_MFPR Rx,ITB_PTE is UNPREDICTABLE.

18. Two HW_MFPR DTB_PTE instructions cannot be issued in consecutive cycles. This implies that more than one instruction may be necessary between the HW_MFPR instructions if dual issue is possible. Similar restrictions apply to the ITB_PTE register.

19. Reading the EXC_SUM and BC_TAG registers require special timing. Refer to Section 3.8.13 and Section 3.10.7 for specific information.

20. DMM errors occurring one cycle before HW_MxPR instructions to the IPTE will NOT stop the TB pointer from incrementing to the next TB entry even though the HW_MxPR instruction will be aborted by the DMM error. This restriction only affects performance and not functionality.

21. PALcode that writes multiple ITB entries must write the entry that maps the address contained in the EXC_ADDR register last.

22. HW_STC instructions cannot be followed, for two cycles, by any load instruction that may miss in the Dcache.

23. Updates to the ASN field of the ICCSR IPR require at least 10 cycles of delay before entering native mode that may reference the ASN during Icache access. If the ASN field is updated in kernel mode via the HWE bit of the ICCSR IPR, it is sufficient that all I-stream references during this time be made to pages with the ASM bit set to avoid use of the ASN.

3.6 Power Up

The table below lists the state of all the IPRs immediately following reset. The table also specifies which IPRs need to be initialized by power-up PALcode.

Table 3–8: IPR Reset State

IPR	Reset State	Comments
ITB_TAG	undefined	
ITB_PTE	undefined	
ICCSR	cleared except ASN,PC0,PC1	Floating point disabled, single issue mode, Pipe mode enabled, ASN = 0, jsr predictions disabled, branch predictions disabled, branch history table disabled, performance counters reset to zero, Perf Cnt0(16b) : Tbtal Issues/2, Perf Cntl(12b) : Dcache Misses
ITB_PTE_TEMP	undefined	
EXC_ADDR	undefined	
SL_RCV	undefined	
ITBZAP	n/a	PALcode must do a itbzap on reset.
ITBASM	n/a	
ITBIS	n/a	
PS	undefined	PALcode must set processor status.
EXC_SUM	undefined	Palcode must clear exception summary and exception register write mask by doing 64 reads.
PAL_BASE	cleared	Cleared on reset.
HIRR	n/a	
SIRR	undefined	PALcode must initialize.
ASTRR	undefined	PALcode must initialize.
HIER	undefined	PALcode must initialize.
SIER	undefined	PALcode must initialize.

Table 3–8 (Cont.): IPR Reset State

IPR	Reset State	Comments
ASTER	undefined	PALcode must initialize.
SL_XMIT	undefined	PALcode must initialize. Appears on external pin.
TB_CTL	undefined	Palcode must select between SP/LP dtb prior to any TB fill.
DTB_PTE	undefined	
DTB_PTE_TEMP	undefined	
MMCSR	undefined	Unlocked on reset.
VA	undefined	Unlocked on reset.
DTBZAP	n/a	PALcode must do a dtbzap on reset.
DTBASM	n/a	
DTBIS	n/a	
BIU_ADDR	undefined	Potentially locked.
BIU_STAT	undefined	Potentially locked.
SL_CLR	undefined	PALcode must initialize.
DC_ADDR	undefined	Potentially locked.
DC_STAT	undefined	Potentially locked.
FILL_ADDR	undefined	Potentially locked.
ABOX_CTL	see comments	[11..0] <- ^x0100 Write buffer enabled, machine checks disabled, correctable read interrupts disabled, Icache stream buffer disabled, Dcache disabled, forced hit mode off.
ALT_MODE	undefined	
CC	undefined	Cycle counter is disabled on reset.
CC_CTL	undefined	
BIU_CTL	see comments	Bcache disabled, parity mode undefined, chip enable asserts during RAM write cycles, Bcache forced-hit mode disabled. BC_PA_DIS field cleared. BAD_TCP cleared. BAD_DP undefined. Note: The Bcache parameters BC RAM read speed, BC RAM write speed, BC write enable control, and BC size are all undetermined on reset and must be initialized before enabling the Bcache.
FILL_SYNDROME	undefined	Potentially locked.
BC_TAG	undefined	Potentially locked.

Table 3–8 (Cont.): IPR Reset State

IPR	Reset State	Comments
PAL_TEMP[31..0]	undefined	

PALcode should execute four jsr call instructions to initialize the jsr stack. This is necessary to insure deterministic behavior for testers. The following code will initialize the stack once the ICCSR [JSE] bit is set.

```
BSR     r1,stk_1        ; push RET PC
stk_1:
BSR     r2,stk_2        ; push RET PC
stk_2:
BSR     r3,stk_3        ; push RET PC
stk_3:
BSR     r4,stk_4        ; push RET PC
stk_4:
```

3.7 TB Miss Flows

This section describes hardware specific details to aid the PALcode programmer in writing ITB and DTB fill routines. These flows were included to highlight trade-offs and restrictions between PAL and hardware. The PALcode source that is released with the 21064 should be consulted before any new flows are written. A working knowledge of the Alpha memory management architecture is assumed.

3.7.1 ITB Miss

When the Ibox encounters an ITB miss it latches the VPC of the target instruction-stream reference in the EXC_ADDR IPR, flushes the pipeline of any instructions following the instruction which caused the ITB miss, waits for any other instructions which may be in progress to complete, enters PALmode, and jumps to the ITB miss PAL entry point. The recommended PALcode sequence for translating the address and filling the ITB is described below.

1. Create some scratch area in the integer register file by writing the contents of a few integer registers to the PAL_TEMP register file.
2. Read the target virtual address from the EXC_ADDR IPR.
3. Fetch the PTE (this may take multiple reads) using a physical-mode HW_LD instruction. If this PTE's valid bit is clear report TNV or ACV as appropriate.

4. Since the Alpha Architecture Handbook states that translation buffers may not contain invalid PTEs, the PTE's valid bit must be explicitly checked by PALcode. Further, since the ITB's PTE RAM does not hold the FOE bit, the PALcode must also explicitly check this condition. If the PTE's valid bit is set and FOE bit is clear, PALcode may fill an ITB entry.
5. Write the original virtual address to the TB_TAG register using HW_MTPR. This writes the TAG into a temp register and not the actual tag field in the ITB.
6. Write the PTE to the ITB_PTE register using HW_MTPR. This HW_MTPR causes both the TAG and PTE fields in the ITB to be written. Note it is not necessary to delay issuing the HW_MTPR to the ITB_PTE after the MTPR to the ITB_TAG is issued.
7. Restore the contents of any modified integer registers to their original state using the HW_MFPR instruction.
8. Restart the instruction stream using the HW_REI instruction.

3.7.2 DTB Miss

When the Abox encounters a DTB miss it latches the referenced virtual address in the VA IPR and other information about the reference in the MMCSR IPR, and locks these registers against further modifications. The Ibox latches the PC of the instruction which generated the reference in the EXC_ADDR register, drains the machine as described above for ITB misses, and jumps to the DTB miss PALcode entry point. Unlike ITB misses, DTB misses may occur while the CPU is executing in PALmode. The recommended PALcode sequence for translating the address and filling the DTB is described below.

1. Create some scratch area in the integer register file by writing the contents of a few integer registers to the PAL_TEMP register file.
2. Read the requested virtual address from the VA IPR. Although the act of reading this register unlocks the VA and MMCSR registers, the MMCSR register only updates when D-stream memory management errors occur. It therefore will retain information about the instruction which generated this DTB miss. This may be useful later.
3. Fetch the PTE (may require multiple reads). If the valid bit of the PTE is clear, a TNV or ACV must be reported unless the instruction which caused the DTB miss was FETCH or FETCH/M. This can be checked via the opcode field of the MMCSR register. If the value in this field is 18 (hex), then a FETCH or FETCH/M instruction caused this DTB miss, and as mandated by the Alpha Architecture Handbook, the subsequent TNV or ACV should NOT be reported. Therefore PALcode should read the value in EXC_ADDR, increment it by four, write this value back to EXC_ADDR, and do a HW_REI.
4. Write the register which holds the contents of the PTE to the TB_CTL IPR. This has the effect of selecting one of the four possible granularity sizes of the DTB for subsequent DTB fill operations, based on the value contained in the granularity hint field of the PTE.
5. Write the original virtual address to the TB_TAG register. This writes the TAG into a temp register and not the actual tag field in the DTB
6. Write TB Size selection to TB_CTL.

7. Wait at least one cycle.
8. Write the PTE to the DTB_PTE register. This HW_MTPR causes both the TAG and PTE fields in the DTB to be written. Note it is not necessary to delay issuing the HW_MTPR to the DTB_PTE after the MTPR to the DTB_TAG is issued.
9. Restore the contents of any modified integer registers.
10. Restart the instruction stream using the HW_REI instruction.

3.8 Internal Processor Registers - IPRs

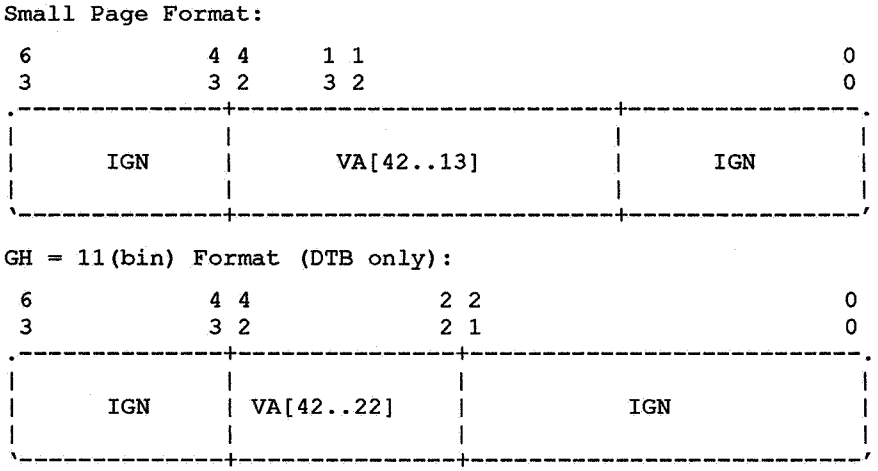
3.8.1 Ibox IPRs

3.8.2 TB_TAG

The TB_TAG register is a write-only register which holds the tag for the next TB update operation in either the ITB or DTB. To insure the integrity of the TB, the tag is actually written to a temporary register and not transferred to the ITB or DTB until the ITB_PTE or DTB_PTE register is written. The entry to be written is chosen at the time of the ITB_PTE or DTB_PTE write operation by a not-last-used algorithm implemented in hardware.

Writing the ITB_TAG register is only performed while in PALmode regardless of the state of the HWE bit in the ICCSR IPR.

Figure 3–4: Translation Buffer Tag (TB_TAG) Registor



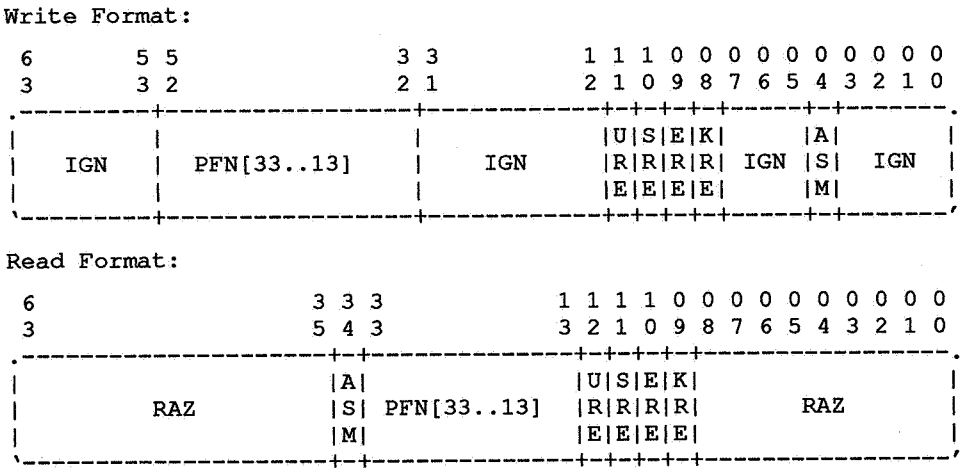
3.8.3 ITB_PTE

The ITB PTE register is a read/write register representing the eight ITB page table entries. The entry to be written is chosen by a not-last-used algorithm implemented in hardware. Writes to the ITB_PTE use the memory format bit positions as described in the Alpha Architecture Handbook with the exception that some fields are ignored.

To insure the integrity of the ITB, the ITB's tag array is updated simultaneously from the internal tag register when the ITB_PTE register is written. Reads of the ITB_PTE require two instructions. First, a read from the ITB_PTE sends the PTE data to the ITB_PTE_TEMP register, then a second instruction reading from the ITB_PTE_TEMP register returns the PTE entry to the register file. Reading or writing the ITB_PTE register increments the TB entry pointer which allows reading the entire set of eight ITB PTE entries.

Reading and writing the ITB_PTE register is only performed while in PALmode regardless of the state of the HWE bit in the ICCSR IPR.

Figure 3–5: ITB_PTE Register

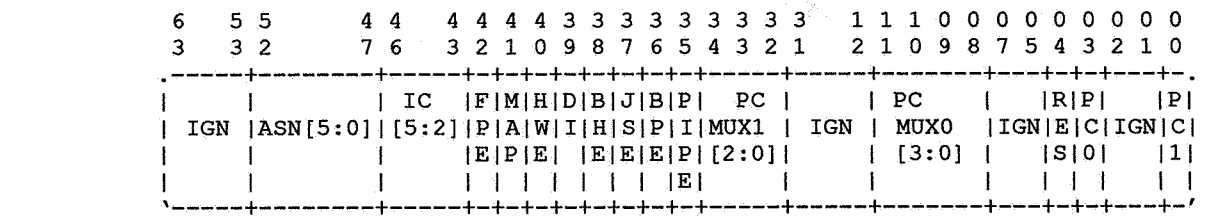


3.8.4 Instruction Cache Control/Status Register - ICCSR

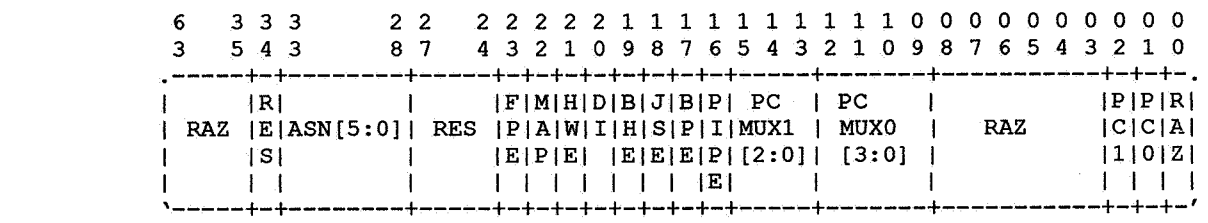
The ICCSR register contains various Ibox hardware enables. The only architecturally defined bit in this register is the FPE, floating point enable, which enables floating point instruction execution. When clear, all floating point instructions generate FEN exceptions. This register is cleared by hardware at reset. The HWE bit allows the special PAL instructions to execute in kernel mode. This bit is intended for diagnostics or operating system alternative PAL routines only. It does not allow access to the ITB registers while not running in PALmode. Therefore, some PALcode flows may require the PALmode environment to execute properly (e.g. ITB fill).

Figure 3–6: ICCSR Register

Write Format:



Read Format:



RES = Reserved

Table 3–9: ICCSR

Field	Type	Description
FPE	RW,0	If set, floating point instructions can be issued. If clear,floating point instructions cause FEN exceptions.
HWE	RW,0	If set, allows the five PALRES instructions to be issued in kernel mode. Use of the HW_MTPR instruction to update the EXC_ADDR IPR while in native mode is restricted to values with bit<0> equal to 0. The combination of native mode and EXC_ADDR<0> equal to one causes UNDEFINED behavior.)
MAP	RW,0	If set, allows super page I-stream memory mapping of VPC<33:13> directly to Physical PC<33:13> essentially bypassing ITB for VPC addresses containing VPC<42:41>= 2. Super page mapping is allowed in Kernel mode only. The ASM bit is always set.
DI	RW,0	If set, enables dual issue.
BHE	RW,0	Used in conjunction with BPE. See table Table 3–9 for programming information.
JSE	RW,0	If set, enables the JSR stack to push return addresses.
BPE	RW,0	Used in conjunction with BHE. See table Table 3–9 for programming information.

Table 3–9 (Cont.): ICCSR

Field	Type	Description
PIPE	RW,0	If clear, causes all hardware interlocked instructions to drain the machine and waits for the write buffer to empty before issuing the next instruction. Examples of instructions that do not cause the pipe to drain include HW_MTPR, HW_REI, conditional branches, and instructions that have a destination register of R31.
PCMUX1	RW,0	See table Table 3–11 for programming information.
PCMUX0	RW,0	See table Table 3–10 for programming information.
PC1	RW	If clear, enables performance counter 1 interrupt request after 2**12 events counted. If set, enables performance counter 1 interrupt request after 2**8 events counted.
PC0	RW	If clear, enables performance counter 0 interrupt request after 2**16 events counted. If set, enables performance counter 0 interrupt request after 2**12 events counted.
ASN	RW	The Address Space Number field is used in conjunction with the Icache to further qualify cache entries and avoid some cache flushes. The ASN is written to the Icache during fill operations and compared with the I-stream data on fetch operations. Mismatches invalidate the fetch without affecting the Icache.
RESERVED	RW	The IC state bits are unused by hardware.

Table 3–10: BHE, BPE Branch Prediction Selection

BPE	BHE	Prediction
0	X	Not Taken
1	0	Sign of Displacement
1	1	Branch History Table.

3.8.4.1 Performance Counters

The performance counters are reset to zero upon powerup, but are otherwise never cleared. They are intended as a means of counting events over a long period of time relative to the event frequency and therefore provide no means of extracting intermediate counter values. Since the counters continuously accumulate selected events despite interrupts being enabled, the first interrupt after selecting a new counter input has an error bound as large as the selected overflow range. In addition, some inputs may overcount events occurring simultaneously with D-stream errors which abort the actual event very late in the pipeline. For example, when counting load instructions, attempts to execute a load resulting in a DTB miss exception will increment the performance counter after the first aborted execution attempt and again after the TB fill routine when the load instruction reissues and completes.

Performance counter interrupts are reported six cycles after the event that caused the counter to overflow. Additional delay may occur before an interrupt is serviced if the processor is executing PALcode which always disables interrupts. In either case, events occurring during the interval between counter overflow and interrupt service are counted toward the next interrupt. Only in the case of a complete counter wraparound while interrupts are disabled will an interrupt be missed.

The six cycles before an interrupt is triggered implies that a maximum of 12 instructions may have completed before the start of the interrupt service routine. In most cases, by examining the possible intervening instructions and the issue rules presented in section 2.9, it is possible to further isolate trigger events. Two cases always provide a more accurate exception PC. When counting Icache misses, no intervening instructions can complete and the exception PC contains the address of the last Icache miss. Branch mispredictions allow a maximum of only 2 instructions to complete before start of the interrupt service routine.

Table 3–11: Performance Counter 0 Input Selection

MUX0[3:0]	Input	Comment
000X	Total Issues / 2	Counts total issues divided by 2, e.g dual issue increments count by 1
001X	Pipeline Dry	Counts cycles where nothing issued due to lack of valid I-stream data. Causes include Icache fill, misprediction, branch delay slots and pipeline drain for exception
010X	Load Instructions	Counts all Load instructions
011X	Pipeline Frozen	Counts cycles where nothing issued due to resource conflict. Refer to section 2.9 for information regarding scheduling and issue rules.
100X	Branch Instructions	Counts all Branch instructions, conditional, unconditional, any JSR, HW_REI
1010	PALmode	Counts cycles while executing in PAL mode
1011	Total cycles	Counts total cycles
110X	Total Non-issues / 2	Counts total non_issues divided by 2, e.g no issue increments count by 1
111X	PERF_CNT_H<0>	Counts external event supplied by pin at selected system clock cycle interval

Table 3–12: Performance Counter 1 Input Selection

MUX1[2:0]	Input	Comment
000	Dcache miss	Counts total Dcache misses

Table 3–12 (Cont.): Performance Counter 1 Input Selection

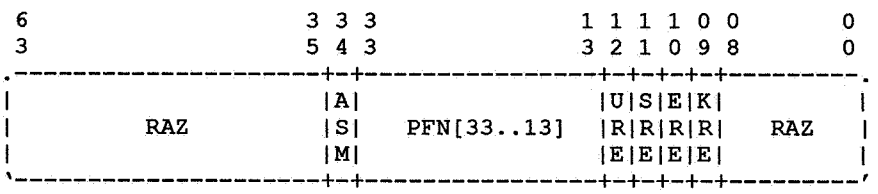
MUX1[2:0]	Input	Comment
001	Icache miss	Counts total Icache misses
010	Dual issues	Counts cycles of Dual issue
011	Branch Mispredicts	Counts both conditional branch mispredictions and JSR or HW_REI mispredictions. Conditional branch mispredictions cost 4 cycles and others cost 5 cycles of dry pipeline delay.
100	FP Instructions	Counts total floating point operate instructions, i.e no FP branch, load, store
101	Integer Operate	Counts integer operate instructions including LDA,LDAH with destination other than R31
110	Store Instructions	Counts total store instructions
111	PERF_CNT_H<1>	Counts external event supplied by pin at selected system clock cycle interval

3.8.5 ITB_PTE_TEMP

The ITB_PTE_TEMP register is a read-only holding register for ITB_PTE read data. Reads of the ITB_PTE require two instructions to return the data to the register file. The first reads the ITB_PTE register to the ITB_PTE_TEMP register. The second returns the ITB_PTE_TEMP register to the integer register file. The ITB_PTE_TEMP register is updated on all ITB accesses, both read and write. A read of the ITB_PTE to the ITB_PTE_TEMP should be followed closely by a read of the ITB_PTE_TEMP to the register file.

Reading the ITB_PTE_TEMP register is only performed while in PALmode regardless of the state of the HWE bit in the ICCSR IPR.

Figure 3–7: ITB_PTE_TEMP Register



3.8.6 Exceptions Address Register (EXC_ADDR)

The EXC_ADDR register is a read/write register used to restart the machine after exceptions or interrupts. The EXC_ADDR register can be read and written by software via the HW_MTPR instruction as well as being written directly by hardware. The HW_REI instruction executes a jump to the address contained in the EXC_ADDR register. The EXC_ADDR register is written by hardware after an exception to provide a return address for PALcode.

The instruction pointed to by the EXC_ADDR register did not complete execution. Since the PC is longword aligned, the lsb of the EXC_ADDR register is used to indicate PALmode to the hardware. When the lsb is clear, the HW_REI instruction executes a jump to native(non-PAL) mode, enabling address translation.

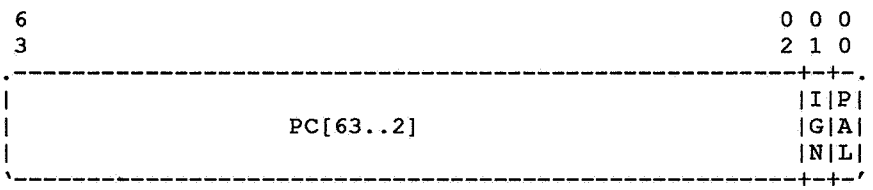
As a special case, CALL_PAL exceptions load the EXC_ADDR with the PC of the instruction following the CALL_PAL. This function allows CALL_PAL service routines to return without needing to increment the value in the EXC_ADDR register.

This feature, however, requires careful treatment in PALcode. Arithmetic traps and machine check exceptions can preempt CALL_PAL exceptions resulting in an incorrect value being saved in the EXC_ADDR register. In the cases of an arithmetic trap or machine check exception, and only in those cases, EXC_ADDR<1> takes on special meaning. PALcode servicing these two exceptions should interpret a zero in EXC_ADDR<1> as indicating that the PC in EXC_ADDR<63:2> is too large by a value of 4bytes and subtract 4 before executing a HW_REI from this address. PALcode should interpret a one in EXC_ADDR<1> as indicating that the PC in EXC_ADDR<63:2> is correct and clear the value of EXC_ADDR<1>. All other PALcode entry points except reset can expect EXC_ADDR<1> to be zero.

This logic allows the following code sequence to conditionally subtract 4 from the address in the EXC_ADDR register without use of an additional register. This code sequence should be present in arithmetic trap and machine check flows only.

```
HW_MFPR Rx, EXC_ADDR    ; read EXC_ADDR into GPR
SUBQ    Rx,#2,Rx        ; subtract 2 causing borrow if <1>=0
BIC     Rx,#2,Rx        ; clear <1>
HW_MTPR Rx, EXC_ADDR    ; write back to EXC_ADDR
```

Figure 3–8: Exception Address Rigister (EXC_ADDR)



3.8.7 SL_CLR

This write-only register clears the serial line interrupt request, the performance counter interrupt requests and the CRD interrupt request. The indicated bit must be written with a zero to clear the selected interrupt source.

Figure 3–9: Clear Serial Line Interrupt Register (SL_CLR)

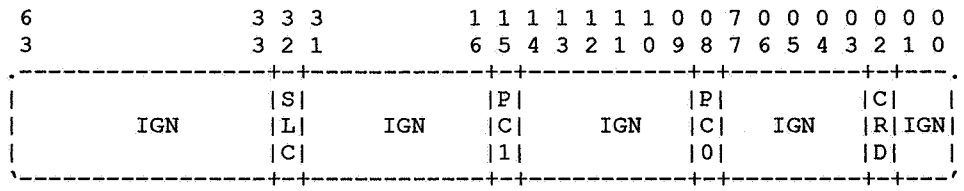


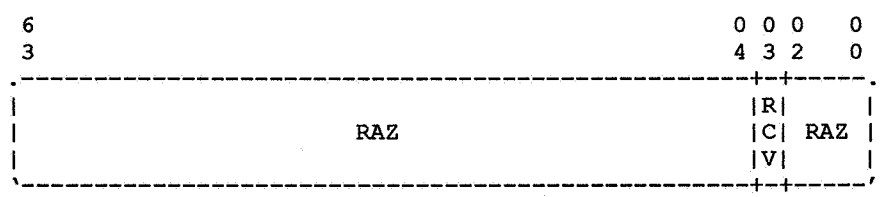
Table 3–13: SL_CLR

Field	Type	Description
CRD	W0C	Clears the correctable read error interrupt request.
PC1	W0C	Clears the performance counter 1 interrupt request.
PC0	W0C	Clears the performance counter 0 interrupt request.
SLC	W0C	Clears the serial line interrupt request.

3.8.8 SL_RCV

The serial line receive register contains a single read-only bit used with the interrupt control registers and the sRomD_h and sRomClk_h pins to provide an on-chip serial line function. The RCV bit is functionally connected to the sRomD_h pin after the Icache is loaded from the external serial ROM. Reading the RCV bit can be used to receive external data one bit at a time under a software timing loop. A serial line interrupt is requested on detection of any transition on the receive line which sets the SL_REQ bit in the HIRR. Using a software timing loop, the RCV bit can be read to receive data one bit at a time. The serial line interrupt can be disabled by clearing the HIER register SL_ENA bit.

Figure 3–10: Serial Line Receive Register (SL_RCV)



3.8.9 ITBZAP

A write of any value to this IPR invalidates all eight ITB entries. It also resets the NLU pointer to its initial state. The ITBZAP register should only be written in PAL mode.

3.8.10 ITBASM

A write of any value to this IPR invalidates all ITB entries in which the ASM bit is equal to zero. The ITBASM register should only be written in PAL mode.

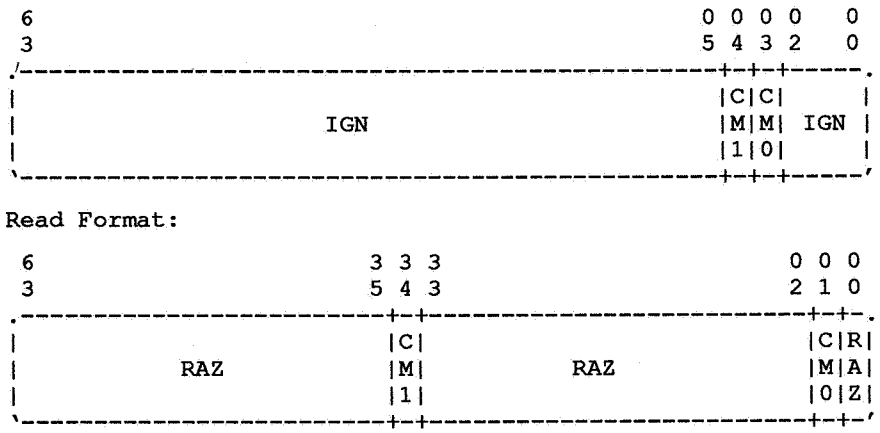
3.8.11 ITBIS

A write of any value to this IPR invalidates all eight ITB entries. It also resets the NLU pointer to its initial state. The ITBIS register should only be written in PAL mode.

3.8.12 PS

The processor status register is a read/write register containing only the current mode bits of the architecturally defined PS.

Figure 3–11: Processor Status Register (PS)



3.8.13 EXC_SUM

The exception summary register records the various types of arithmetic traps that have occurred since the last time the EXC_SUM was written (cleared). When the result of an arithmetic operation produces an arithmetic trap, the corresponding EXC_SUM bit is set.

In addition, the register containing the result of that operation is recorded in the exception register write mask IPR, as a single bit in a 64-bit field specifying registers F31-F0 and I31-I0. This IPR is visible only through the EXC_SUM register. The EXC_SUM register provides a one-bit window to the exception register write mask. Each read to the EXC_SUM shifts one bit in order F31-F0 then I31-I0. The read also clears the corresponding bit. Therefore, the EXC_SUM must be read 64 times to extract the complete mask and clear the entire register.

Any write to EXC_SUM clears bits [8..2] and does not affect the write mask.

The write mask register bit clears three cycles after a read. Therefore, code intended to read the register must allow at least three cycles between reads to allow the clear and shift operation to complete in order to insure reading successive bits.

Figure 3–12: Exception Summary Register (EXC_SUM)

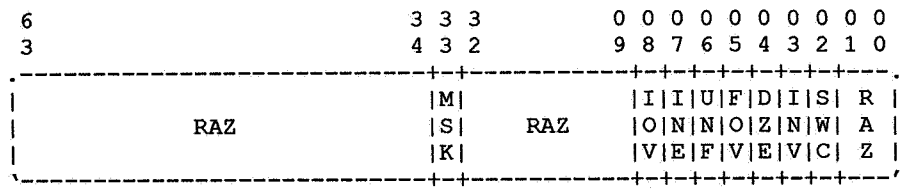


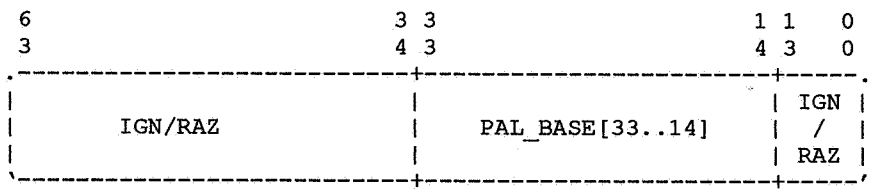
Table 3–14: EXC_SUM

Field	Type	Description
SWC	WA	Indicates Software Completion possible. The bit is set after a floating point instruction containing the /S modifier completes with an arithmetic trap and all previous floating point instructions that trapped since the last MTPR EXC_SUM also contained the /S modifier. The SWC bit is cleared whenever a floating point instruction without the /S modifier completes with an arithmetic trap. The bit remains cleared regardless of additional arithmetic traps until the register is written via an MTPR instruction. The bit is always cleared upon any MTPR write to the EXC_SUM register.
INV	WA	Indicates Invalid Operation.
DZE	WA	Indicates Divide by Zero.
FOV	WA	Indicates Floating Point Overflow.
UNF	WA	Indicates Floating Point Underflow.
INE	WA	Indicates Floating Inexact Error.
IOV	WA	Indicates Fbox Convert to Integer Overflow or Integer Arithmetic Overflow.
MSK	RC	Exception Register Write Mask IPR Window.

3.8.14 PAL Base Address Register (PAL_BASE)

The PAL base register is a read/write register containing the base address for PALcode. This register is cleared by hardware at reset.

Figure 3–13: PAL Base Register (PAL_BASE)



3.8.15 HIRR

The Hardware Interrupt Request Register is a read-only register providing a record of all currently outstanding interrupt requests and summary bits at the time of the read. For each bit of the HIRR [5:0] there is a corresponding bit of the HIER (Hardware Interrupt Enable Register) that must be set to request an interrupt. In addition to returning the status of the hardware interrupt requests, a read of the HIRR returns the state of the software interrupt and AST requests. Note that a read of the HIRR may return a value of zero if the hardware interrupt was released before the read (passive release). The register guarantees that the HWR bit reflects the status as shown by the HIRR bits. All interrupt requests are blocked while executing in PALmode.

See section Section 2.3.3 for description of interrupt logic.

Figure 3–14: Hardware Interrupt Request Register (HIRR)

Read Format:

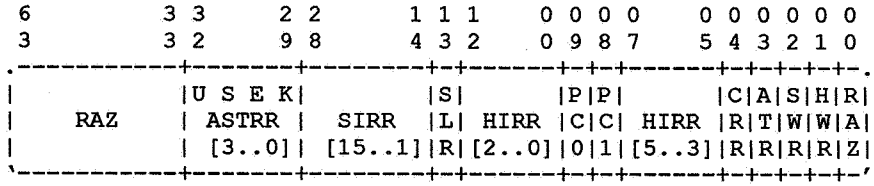


Table 3–15: HIRR

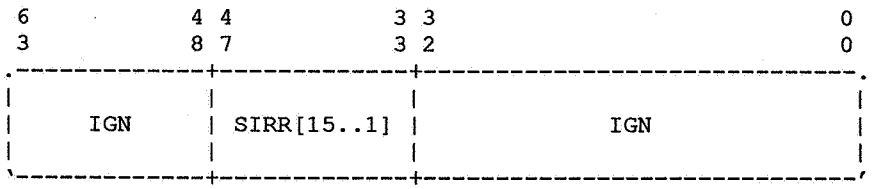
Field	Type	Description
HWR	RO	Is set if any hardware interrupt request and corresponding enable is set
SWR	RO	Is set if any software interrupt request and corresponding enable is set
ATR	RO	Is set if any AST request and corresponding enable is set. This bit also requires that the processor mode be equal to or higher than the request mode. SIER[2] must be set to allow AST interrupt requests.
HIRR[5..0]	RO	Corresponds to pins Irq_h[5..0].
SIRR[15..1]	RO	Corresponds to software interrupt request 15 thru 1.
ASTRR[3..0]	RO	Corresponds to AST request three thru zero (USEK).
PC1	RO	Performance counter 1 interrupt request.
PC0	RO	Performance counter 0 interrupt request.
SLR	RO	Serial line interrupt request. (See also SL_RCV, SL_XMIT, and SL_CLR.)
CRR	RO	CRD correctable read error interrupt request. This interrupt request is cleared via the SL_CLR register.

3.8.16 SIRR

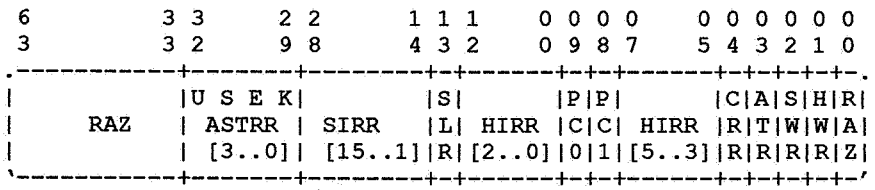
The Software Interrupt Request Register is a read/write register used to control software interrupt requests. For each bit of the SIRR there is a corresponding bit of the SIER (Software Interrupt Enable Register) that must be set to request an interrupt. Reads of the SIRR return the complete set of interrupt request registers and summary bits, see the HIRR Table 3–15 for details. All interrupt requests are blocked while executing in PALmode.

Figure 3–15: Software Interrupt Request Register (SIRR)

Write Format:



Read Format:

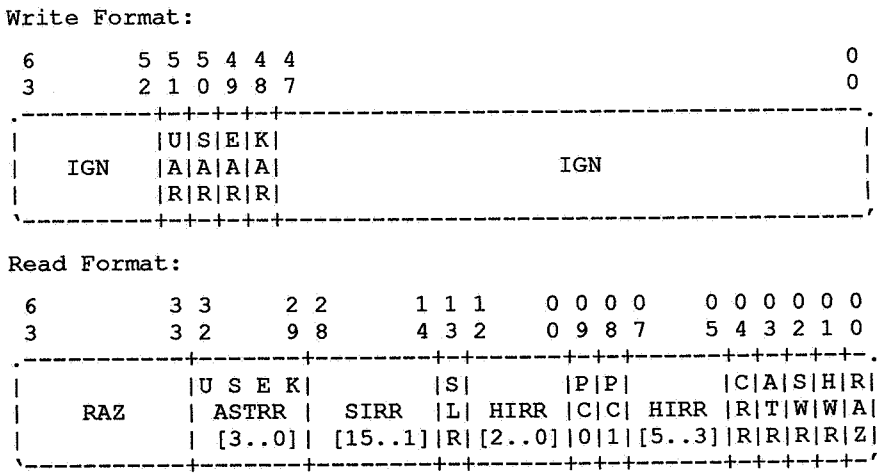


3.8.17 ASTRR

The Asynchronous Trap Request Register is a read/write register. It contains bits to request AST interrupts in each of the processor modes. In order to generate an AST interrupt, the corresponding enable bit in the ASTER must be set and the processor must be in the selected processor mode or higher privilege as described by the current value of the PS CM bits. AST interrupts are enabled if the SIER[2] is set. This provides a mechanism to lock out AST requests over certain IPL levels.

All interrupt requests are blocked while executing in PALmode. Reads of the ASTRR return the complete set of interrupt request registers and summary bits, see the HIRR Table 3–15 for details.

Figure 3–16: Asynchronous Trap Request Register (ASTRR)



3.8.18 Hardware Interrupt Enable - HIER

The Hardware Interrupt Enable Register is a read/write register. It is used to enable corresponding bits of the HIRR requesting interrupt. The PC0, PC1, SLE and CRE bits of this register enable the performance counters, serial line and correctable read interrupts. There is a one-to-one correspondence between the interrupt requests and enable bits, as with the reads of the interrupt request IPRs, reads of the HIER return the complete set of interrupt enable registers, see the HIRR Table 3–15 for details.

Figure 3–17: Hardware Interrupt Enable Register (HIER)

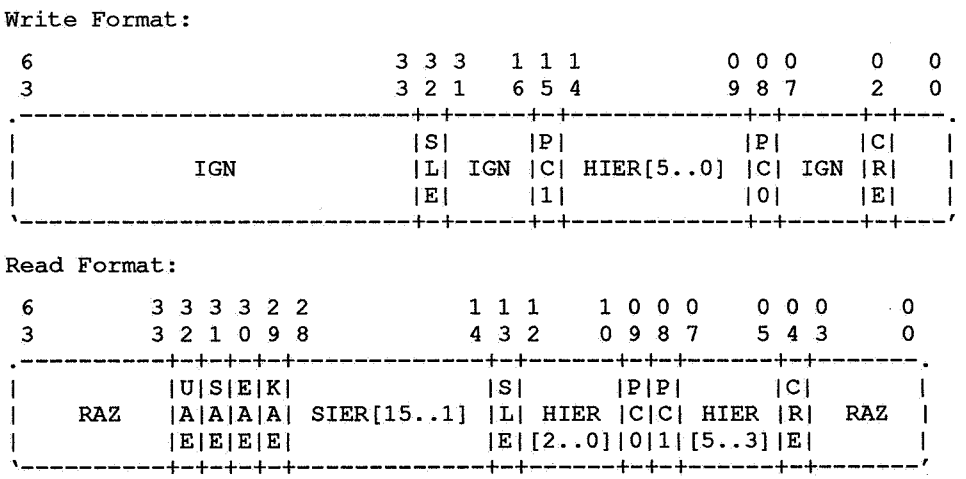


Table 3–16: HIER

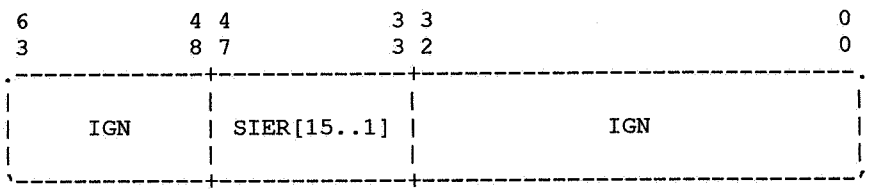
Field	Type	Description
CRR	RO	CRD correctable read error interrupt enable.
HIER[5..0]	RO	Corresponds to pins Irq_h[5..0].
SIER[15..1]	RO	Corresponds to software interrupt requests 15 thru 1.
ASTRR[3..0]	RO	Corresponds to AST enable three thru zero (USEK).
PC1	RO	Performance counter 1 interrupt enable.
PC0	RO	Performance counter 0 interrupt enable.
SLE	RO	Serial line interrupt enable. (See also SL_RCV, SL_XMIT, and SL_CLR).
CRE	RO	CRD correctable read error interrupt enable. This interrupt request is cleared via the SL_CLR register.

3.8.19 SIER

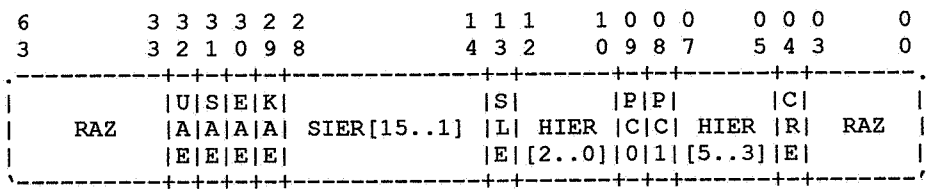
The Software Interrupt Enable Register is a read/write register. It is used to enable corresponding bits of the SIRR requesting interrupts. There is a one-to-one correspondence between the interrupt requests and enable bits, as with the reads of the interrupt request IPRs, reads of the SIER return the complete set of interrupt enable registers, see the HIRR Table 3–15 for details.

Figure 3–18: Software Interrupt Enable Register (SIER)

Write Format:



Read Format:

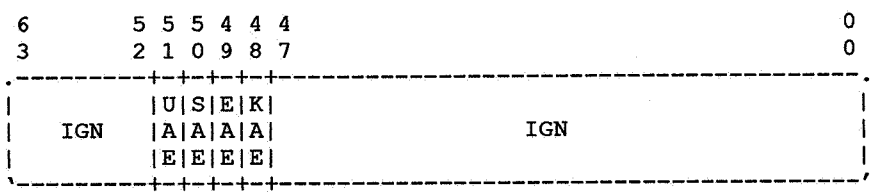


3.8.20 ASTER

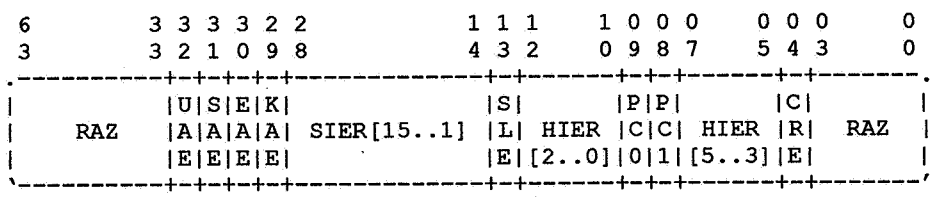
The AST Interrupt Enable Register is a read/write register. It is used to enable corresponding bits of the ASTRR requesting interrupts. There is a one-to-one correspondence between the interrupt requests and enable bits, as with the reads of the interrupt request IPRs, reads of the ASTER return the complete set of interrupt enable registers, see the HIRR Table 3–15 for details.

Figure 3–19: AST Interrupt Enable Register (ASTER)

Write Format:



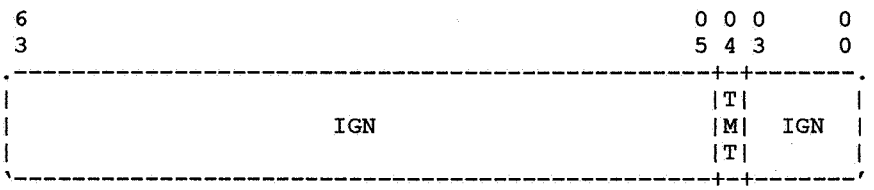
Read Format:



3.8.21 SL_XMIT

The Serial Line Transmit register contains a single write-only bit used with the interrupt control registers and the sRomD_h and sRomClk_h pins to provide an on-chip serial line function. The TMT bit is functionally connected to the sRomClk_h pin after the Icache is loaded from the external serial ROM. Writing the TMT bit can be used to transmit data off chip one bit at a time under a software timing loop.

Figure 3–20: Serial Line Transmit Register (SL_XMIT)

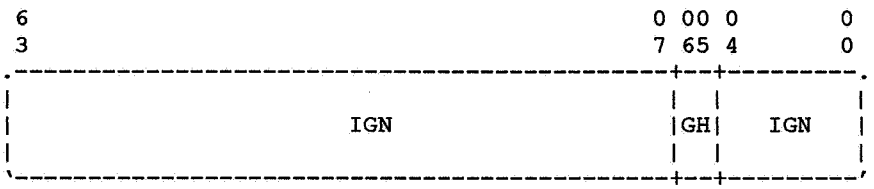


3.9 Abox IPRs

3.9.1 TB_CTL

The granularity hint (GH) field selects between the 21064 TB page mapping sizes. There are two sizes in the ITB and all four sizes in the DTB. When only two sizes are provided, the large-page-select (GH=11(bin)) field selects the largest mapping size (512 * 8Kbytes) and all other values select the smallest (8Kbyte) size. The GH field affects both reads and writes to the ITB and DTB.

Figure 3–21: TB_CTL Register



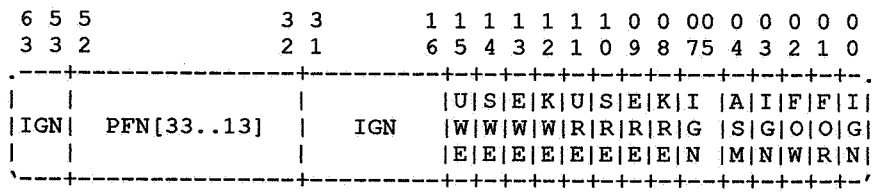
3.9.2 DTB_PTE

The DTB PTE register is a read/write register representing the 32-entry small-page and 4-entry large-page DTB page table entries. The entry to be written is chosen by a not-last-used (NLU) algorithm implemented in hardware and the value in the TB_CTL register. Writes to the DTB_PTE use the memory format bit positions as described in the Alpha Architecture Handbook with the exception that some fields are ignored. In particular the valid bit is not represented in hardware.

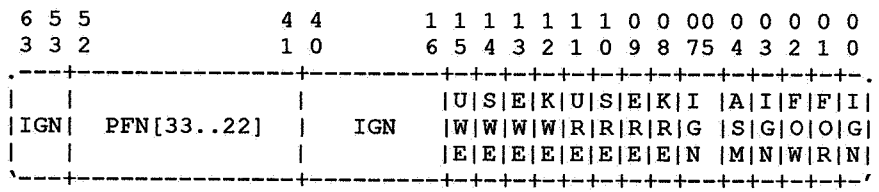
To insure the integrity of the DTBs, the DTB's tag array is updated simultaneously from the internal tag register when the DTB_PTE register is written. Reads of the DTB_PTE require two instructions. First, a read from the DTB_PTE sends the PTE data to the DTB_PTE_TEMP register, then a second instruction reading from the DTB_PTE_TEMP register returns the PTE entry to the register file. Reading or writing the DTB_PTE register increments the TB entry pointer of the DTB indicated by the TB_CTL IPR which allows reading the entire set of DTB PTE entries.

Figure 3–22: DTB_PTE Register

Small Page Format:



Large Page Format:

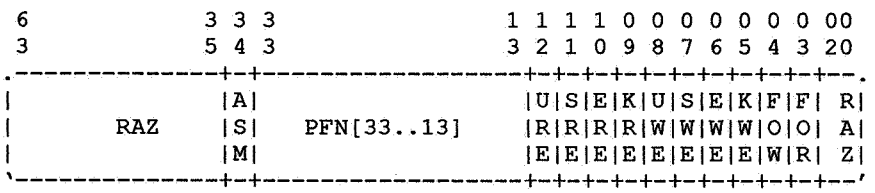


3.9.3 DTB_PTE_TEMP

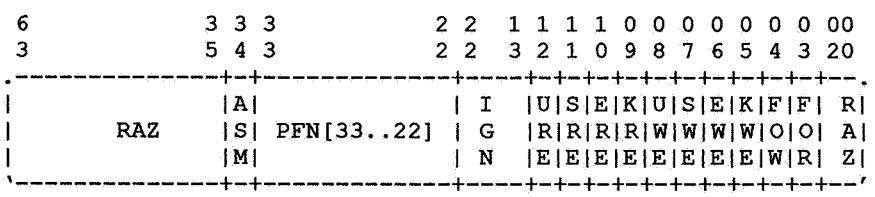
The DTB_PTE_TEMP register is a read-only holding register for DTB_PTE read data. Reads of the DTB_PTE require two instructions to return the data to the register file. The first reads the DTB_PTE register to the DTB_PTE_TEMP register. The second returns the DTB_PTE_TEMP register to the integer register file.

Figure 3–23: DTB_PTE_TEMP Register

Small Page Format:



Large Page Format:



3.9.4 MM_CSR

When D-stream faults occur the information about the fault is latched and saved in the MM_CSR register. The VA and MM_CSR registers are locked against further updates until software reads the VA register. Palcode must explicitly unlock this register whenever its entry point was higher in priority than a DTB miss. MM_CSR bits are only modified by hardware when the register is not locked and a memory management error or a DTB miss occurs. The MM_CSR is unlocked after reset.

Figure 3–24: MM_CSR Register

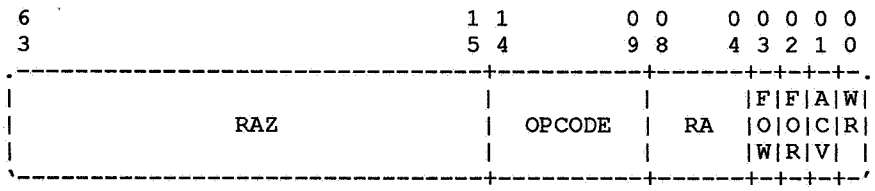


Table 3–17: MM_CSR

Field	Type	Description
WR	RO	Set if reference which caused error was a write.
ACV	RO	Set if reference caused an access violation.
FOR	RO	Set if reference was a read and the PTE's FOR bit was set.
FOW	RO	Set if reference was a write and the PTE's FOW bit was set.
RA	RO	Ra field of the faulting instruction.
OPCODE	RO	Opcode field of the faulting instruction.

3.9.5 Virtual Address Register (VA)

When D-stream faults or DTB misses occur the effective virtual address associated with the fault or miss is latched in the read-only VA register. The VA and MM_CSR registers are locked against further updates until software reads the VA register. The VA IPR is unlocked after reset. Palcode must explicitly unlock this register whenever its entry point was higher in priority than a DTB miss.

3.9.6 DTBZAP

A write of any value to this IPR invalidates all 32 small-page and four large-page DTB entries. It also resets the NLU pointer to its initial state.

3.9.7 DTBASM

A write of any value to this IPR invalidates all 32 small-page and 4 large-page DTB entries in which the ASM bit is equal to zero.

3.9.8 DTBIS

If the virtual address in the RB field is mapped in either the small-page or large-page DTB then those entries are invalidated.

3.9.9 FLUSH_IC

A write of any value to this pseudo-IPR flushes the entire instruction cache.

3.9.10 FLUSH_IC_ASM

A write of any value to this pseudo-IPR invalidates all Icache blocks in which the ASM bit is clear.

3.9.11 Abox Control Register (ABOX_CTL)

Figure 3–25: Abox Control Register (ABOX_CTL)

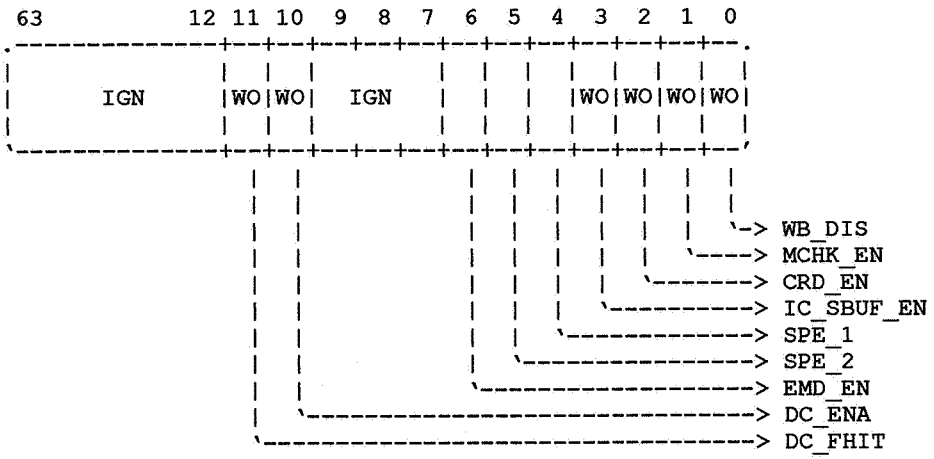


Table 3–18: Abox Control Register

Field	Type	Description
WB_DIS	WO,0	Write Buffer unload Disable. When set, this bit prevents the write buffer from sending write data to the BIU. It should be set for diagnostics only.
MCHK_EN	WO,0	Machine Check Enable. When this bit is set the Abox generates a machine check when errors which are not correctable by the hardware are encountered. When this bit is cleared, uncorrectable errors do not cause a machine check, but the BIU_STAT, DC_STAT, BIU_ADDR, FILL_ADDR and DC_ADDR registers are updated and locked when the errors occur.
CRD_EN	WO,0	Corrected read data interrupt enable. When this bit is set the Abox generates an interrupt request whenever a pin bus transaction is terminated with a cAck_h code of SOFT_ERROR.
IC_SBUF_EN	WO,0	Icache stream buffer enable. When set, this bit enables operation of a single entry Icache stream buffer.

Table 3–18 (Cont.): Abox Control Register

Field	Type	Description
SPE_1	WO,0	This bit, when set, enables one-to-one super page mapping of D-stream virtual addresses with VA<33:13> directly to physical addresses PA<33:13>, if virtual address bits VA<42:41> = 2. Virtual address bits VA(<)40:34> are ignored in this translation. Access is only allowed in kernel mode.
SPE_2	WO,0	This bit, when set, enables one-to-one super page mapping of D-stream virtual addresses with VA<42:30> = 1FFE(Hex) to physical addresses with PA<33:30> = 0(Hex). Access is only allowed in kernel mode.
EMD_EN	WO,0	Limited hardware support is provided for big endian data formats via bit <6><LITERAL> of the ABOX_CTL register. This bit, when set, inverts physical address bit <LITERAL><2> for all D-stream references. It is intended that chip endian mode be selected during initialization PALcode only.
DC_EN	WO,0	Dcache enable. When clear, this bit disables and flushes the Dcache. When set, this bit enables the Dcache.
DC_FHIT	WO,0	Dcache force hit. When set, this bit forces all D-stream references to hit in the Dcache. This bit takes precedence over DC_EN, i.e. when DC_FHIT is set and DC_EN is clear all D-stream references hit in the Dcache.

3.9.12 ALT_MODE

ALT_MODE is a write-only IPR. The AM field specifies the alternate processor mode used by HW_LD and HW_ST instructions which have their ALT bit (bit 14) set.

Figure 3–26: Alternate Processor Mode Register (ALT_MODE)

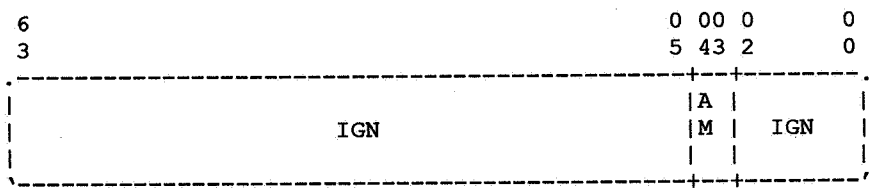


Table 3–19: ALT Mode

ALT_ MODE[4..3]	Mode
0 0	Kernel
0 1	Executive
1 0	Supervisor
1 1	User

3.9.13 Cycle Counter (CC)

The 21064 supports a cycle counter as described in the Alpha Architecture Handbook. This counter, when enabled, increments once each CPU cycle. HW_MTPR Rn,CC writes CC[63..32] with the value held in Rn[63..32], and CC[31..0] are not changed. This register is read by the RPCC instruction defined in the Alpha Architecture Handbook.

3.9.14 Cycle Counter Control Register (CC_CTL)

HW_MTPR Rn,CC_CTL writes CC[31..0] with the value held in Rn[31..0], and CC[63..32] are not changed. CC[3..0] must be written with zero. If Rn[32] is set then the counter is enabled, otherwise the counter is disabled. CC_CTL is a write-only IPR.

3.9.15 Bus Interface Unit Control Register (BIU_CTL)

Figure 3–27: Bus Interface Unit Control Register (BIU_CTL)

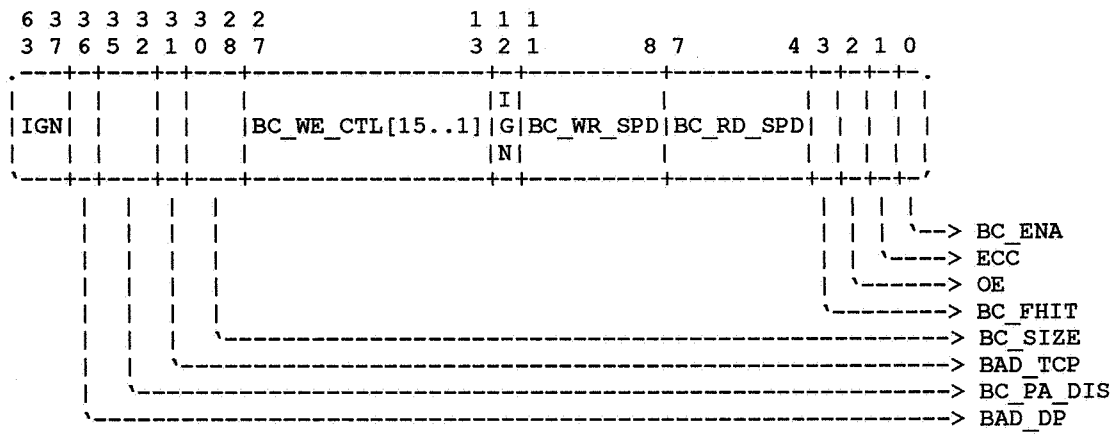


Table 3–20: BIU Control Register

Field	Type	Description
BC_EN	WO,0	External cache enable. When clear, this bit disables the external cache. When the external cache is disabled the BIU does not probe the external cache tag store for read and write references; it launches a request on cReq_h immediately.
ECC	WO	When this bit is set the 21064 generates/expects ECC on the check_h pins. When this bit is clear the chip generates/expects parity on four of the check_h pins.
OE	WO,0	When this bit is set the 21064 does not assert its chip enable pins during RAM write cycles, thus enabling these pins to be connected to the output enable pins of the cache RAMs.
BC_FHIT	WO,0	External cache force hit. When this bit is set and BC_EN is also set, all pin bus READ_BLOCK and WRITE_BLOCK transactions are forced to hit in the external cache. Tag and tag control parity are ignored when the BIU operates in this mode. BC_EN takes precedence over BC_FHIT. When BC_EN is clear and BC_FHIT is set no tag probes occur and external requests are directed to the cReq_h pins. Note that the BC_PA_DIS field takes precedence over the BC_FHIT bit.
BC_RD_SPD	WO	External cache read speed. This field indicates to the BIU the read access time of the RAMs used to implement the off-chip external cache, measured in CPU cycles. It should be written with a value equal to one less the read access time of the external cache RAMs. Access times for reads must be in the range 16..3 CPU cycles, which means the values for the BC_RD_SPD field are in the range of 15..2. BC_RD_SPD are not initialized on reset and must be explicitly written before enabling the external cache.
BC_WR_SPD	WO	External cache write speed. This field indicates to the BIU the write cycle time of the RAMs used to implement the off-chip external cache, measured in CPU cycles. It should be written with a value equal to one less the write cycle time of the external cache RAMs. Access times for writes must be in the range 16..2 CPU cycles, which means the values for the BC_RD_SPD field are in the range of 15..1. BC_WR_SPD are not initialized on reset and must be explicitly written before enabling the external cache.

Table 3–20 (Cont.): BIU Control Register

Field	Type	Description
BC_WE_CTL	WO	External cache write enable control. This field is used to control the timing of the write enable and chip enable pins during writes into the data and tag control RAMs. It consists of 15 bits, where each bit determines the value placed on the write enable and chip enable pins during a given CPU cycle of the RAM write access. When a given bit of BC_WE_CTL is set, the write enable and chip enable pins are asserted during the corresponding CPU cycle of the RAM access. BC_WE_CTL[0] (bit 13 in BIU_CTL) corresponds to the second cycle of the write access, BC_WE_CTL[1] (bit 14 in BIU_CTL) to the third CPU cycle, and so on. The write enable pins will never be asserted in the first CPU cycle of a RAM write access. Unused bits in the BC_WE_CTL field must be written with zeros. BC_WE_CTL is not initialized on reset and must be explicitly written before enabling the external cache.
BC_SIZE	WO	This field is used to indicate the size of the external cache. BC_SIZE is not initialized on reset and must be explicitly written before enabling the external cache. See Table 3–19 for the encodings.
BAD_TCP	WO,0	When set, BAD_TCP causes the 21064 to write bad parity into the tag control RAM whenever it does a fast external RAM write.
BC_PA_DIS	WO	This 4-bit field may be used to prevent the CPU chip from using the external cache to service reads and writes based upon the quadrant of physical address space which they reference. The correspondence between this bit field and the physical address space is shown in Table 3–20. When a read or write reference is presented to the BIU the values of BC_PA_DIS, BC_ENA and physical address bits [33:32] together determine whether to attempt to use the external cache to satisfy the reference. If the external cache is not to be used for a given reference the BIU does not probe the tag store, and makes the appropriate system request immediately. The value of BC_PA_DIS has NO impact on which portions of the physical address space may be cached in the primary caches. System components control this via the RDACK field of the pin bus. BC_PA_DIS are not initialized by reset.
BAD_DP	WO	When set, BAD_DP causes the 21064 to invert the value placed on bits [0],[7],[14] and [21] of the check_h[27..0] field during off-chip writes. This produces bad parity when the 21064 is in parity mode, and bad check bit codes when it is in ECC mode.

Table 3–21: BC_SIZE

BC_SIZE	Size
0 0 0	128 Kbytes
0 0 1	256 Kbytes
0 1 0	512 Kbytes
0 1 1	1 Mbytes
1 0 0	2 Mbytes
1 0 1	4 Mbytes
1 1 0	8 Mbytes

Table 3–22: BC_PA_DIS

BIU_CTL bits	Physical Address
[32]	PA[33..32] = 0
[33]	PA[33..32] = 1
[34]	PA[33..32] = 2
[35]	PA[33..32] = 3

3.10 PAL_TEMPs

The CPU chip contains 32 registers which are accessible via HW_MxPR instructions. These registers provide temporary storage for PALcode.

3.10.1 DC_STAT

The DC_STAT is a read-only IPR.

Overview:

When an external ECC or parity error is recognized during a primary cache fill operation, the DC_STAT register is locked against further updates. In the event that the cache fill was due to D-stream activity the contents of this register may be used by PAL code in conjunction with information latched elsewhere (see Section 3.12) to recover from some single-bit ECC errors. DC_STAT is unlocked when DC_ADDR is read.

Figure 3–28: Data Cache Status Register (DC_STAT)

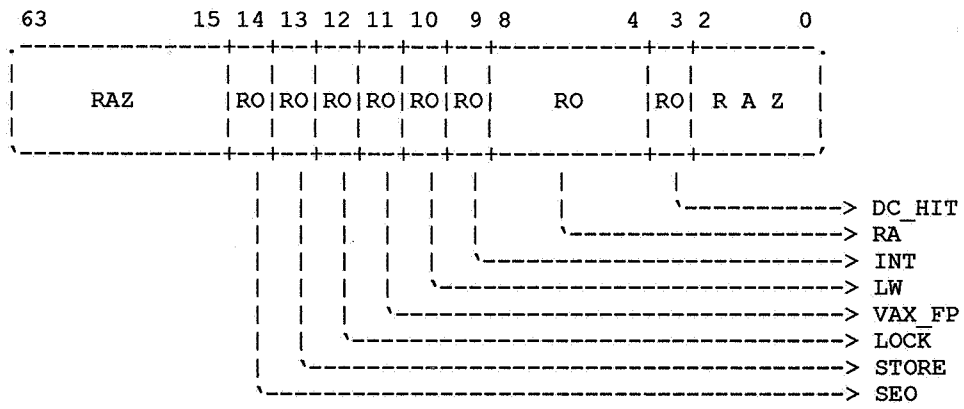


Table 3–23: Dcache Status Register

Field	Type	Description
DC_HIT	RO	This bit indicates whether the last load or store instruction processed by the Abox hit, (DC_HIT set) or missed, (DC_HIT clear) the Dcache. Loads that miss the Dcache may be completed without requiring external reads. e.g. pending fill or pending store hits.
SEO	RO	Second Error Occurred. Set when an error which would normally lock the DC_STAT register occurs while the DC_STAT register is already locked.

The following bits are only meaningful if the FILL_ECC or FILL_DPERR bit in the BIU_STAT register is set.

Table 3–24: Dcache STAT Error Modifiers

Field	Type	Description
RA	RO	The Ra field of the instruction which resulted in the error.
INT	RO	When set, indicates an integer load or store.
LW	RO	When set, indicates that the data length of the load or store was longword.
VAX_FP	RO	When INT is clear, this bit is set to indicate that a VAX floating point format load or store caused the error.
LOCK	RO	This bit is set to indicate that the error stemmed from a LDLL, LDQL, STLC, or STQC instruction.
STORE	RO	This bit is set to indicate that the error stemmed from a store instruction.

3.10.2 DC_ADDR

DC_ADDR is a pseudo-register used for unlocking DC_STAT. DC_STAT and DC_ADDR are unlocked when DC_ADDR is read.

3.10.3 BIU_STAT

BIU_STAT is a read-only IPR.

When one of BIU_HERR, BIU_SERR, BC_TPERR or BC_TCPERR is set, BIU_STAT[6..0] are locked against further updates, and the address associated with the error is latched and locked in the BIU_ADDR register. BIU_STAT[6..0] and BIU_ADDR are also spuriously locked when FILL_ECC or FILL_DPERR is set. BIU_STAT[7..0] and BIU_ADDR are unlocked when the BIU_ADDR register is read.

When FILL_ECC or FILL_DPERR is set, BIU_STAT[13..8] are locked against further updates, and the address associated with the error is latched and locked in the FILL_ADDR register. BIU_STAT[14..8] and FILL_ADDR are unlocked when the FILL_ADDR register is read.

This register is not unlocked or cleared by reset and needs to be explicitly cleared by PALcode.

Figure 3–29: Bus Interface Unit Status Register (BIU_STAT)

```
 63        14 13 12 11 10  9  8  7 6    5  4  3  2  1  0
.---------+--+------+--+--+--+--+--+--------+--+--+--+--.
|         |  |      |  |  | R|  |  |        |  |  |  |  |
|   RAZ   |RO|  RO  |RO|RO| A|RO|RO|   RO   |RO|RO|RO|RO|
|         |  |      |  |  | Z|  |  |        |  |  |  |  |
`---------+--+------+--+--+--+--+--+--------+--+--+--+--'
           |    |    |  |     |  |    |     |  |  |  |
           |    |    |  |     |  |    |     |  |  |  `-> BIU_HERR
           |    |    |  |     |  |    |     |  |  `----> BIU_SERR
           |    |    |  |     |  |    |     |  `-------> BC_TPERR
           |    |    |  |     |  |    |     `----------> BC_TCPERR
           |    |    |  |     |  |    `----------------> BIU_CMD
           |    |    |  |     |  `---------------------> BIU_SEO
           |    |    |  |     `------------------------> FILL_ECC
           |    |    |  `------------------------------> FILL_DPERR
           |    |    `---------------------------------> FILL_IRD
           |    `--------------------------------------> FILL_QW
           `-------------------------------------------> FILL_SEO
```

Table 3–25: BIU STAT

Field	Type	Description
BIU_HERR	RO	This bit, when set, indicates that an external cycle was terminated with the cAck_h pins indicating HARD_ERROR.

Table 3–25 (Cont.): BIU STAT

Field	Type	Description
BIU_SERR	RO	This bit, when set, indicates that an external cycle was terminated with the cAck_h pins indicating SOFT_ERROR.
BC_TPERR	RO	This bit, when set, indicates that a external cache tag probe encountered bad parity in the tag address RAM.
BC_TCPERR	RO	This bit, when set, indicates that a external cache tag probe encountered bad parity in the tag control RAM.
BIU_CMD	RO	This field latches the cycle type on the cReq_h pins when a BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR error occurs.
BIU_SEO	RO	This bit, when set, indicates that an external cycle was terminated with the cAck_h pins indicating HARD_ERROR or that a an external cache tag probe encountered bad parity in the tag address RAM or the tag control RAM while one of BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR was already set.
FILL_ECC	RO	ECC error. This bit, when set, indicates that primary cache fill data received from outside the CPU chip contained an ECC error.
FILL_DPERR	RO	Fill Parity Error. This bit when set, indicates that the BIU received data with a parity error from outside the CPU chip while performing either a Dcache or Icache fill. FILL_DPERR is only meaningful when the CPU chip is in parity mode, as opposed to ECC mode.
FILL_IRD	RO	This bit is only meaningful when either FILL_ECC or FILL_DPERR is set. FILL_IRD is set to indicate that the error which caused FILL_ECC or FILL_DPERR to set occurred during an Icache fill and clear to indicate that the error occurred during a Dcache fill.
FILL_QW	RO	This field is only meaningful when either FILL_ECC or FILL_DPERR is set. FILL_QW identifies the quadword within the hexaword primary cache fill block which caused the error. It can be used together with FILL_ADDR[33..5] to get the complete physical address of the bad quadword.
FILL_SEO	RO	This bit, when set, indicates that a primary cache fill operation resulted in either an uncorrectable ECC error or in a parity error while FILL_ECC or FILL_DPERR was already set.

3.10.4 BIU_ADDR

This read-only register contains the physical address associated with errors reported by BIU_STAT[7..0]. Its contents are meaningful only when one of BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR are set. Reads of BIU_ADDR unlock both BIU_ADDR and BIU_STAT[7..0].

BIU_ADDR[33..5] contain the values of adr_h[33..5] associated with the pin bus transaction which resulted in the error indicated in BIU_STAT[7..0].

If the BIU_CMD field of the BIU_STAT register indicates that the transaction which received the error was READ_BLOCK or LDx/L, then BIU_STAT[4..2] are UNPREDICTABLE. If the BIU_CMD field of the BIU_STAT register encodes any pin bus command other than READ_BLOCK or LDx/L, then BIU_ADDR[4..2] will contain zeros. BIU_ADDR[63..34] and BIU_ADDR[1..0] always read as zero.

3.10.5 FILL_ADDR

This read-only register contains the physical address associated with errors reported by BIU_STAT[14..8]. Its contents are meaningful only when FILL_ECC or FILL_DPERR is set. Reads of FILL_ADDR unlock FILL_ADDR, BIU_STAT[14..8] and FILL_SYNDROME.

FILL_ADDR[33..5] identify the 32-byte cache block which the CPU was attempting to read when the error occurred.

If the FILL_IRD bit of the BIU_STAT register is clear, indicating that the error occurred during a D-stream cache fill, then FILL_ADDR[4..2] contain bits [4..2] of the physical address generated by the load instruction which triggered the cache fill. If FILL_IRD is set, then FILL_ADDR[4..2] are UNPREDICTABLE. FILL_ADDR[63..34] and FILL_ADDR[1..0] read as zero.

3.10.6 FILL_SYNDROME

The FILL_SYNDROME register is a 14-bit read-only register.

If the chip is in ECC mode and an ECC error is recognized during a primary cache fill operation, the syndrome bits associated with the bad quadword are locked in the FILL_SYNDROME register. FILL_SYNDROME[6..0] contain the syndrome associated with the lower longword of the quadword, and FILL_SYNDROME[13..7] contain the syndrome associated with the higher longword of the quadword. A syndrome value of zero means that no errors where found in the associated longword. See Table 3–24 for a list of syndromes associated with correctable single-bit errors. The FILL_SYNDROME register is unlocked when the FILL_ADDR register is read.

If the chip is in parity mode and a parity error is recognized during a primary cache fill operation, the FILL_SYNDROME register indicates which of the longwords in the quadword got bad parity. FILL_SYNDROME[0] is set to indicate that the low longword was corrupted, and FILL_SYNDROME[7] is set to indicate that the high longword was corrupted. FILL_SYNDROME[13..8] and [6..1] are RAZ in parity mode.

Figure 3–30: FILL_SYNDROME Register

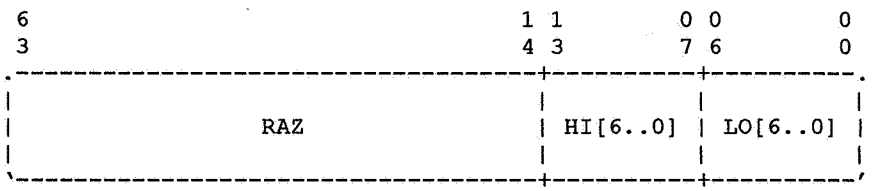


Table 3–26: Syndromes for Single-Bit Errors

Data Bit	Syndrome(Hex)	Check Bit	Syndrome(Hex)
00	4F	00	01
01	4A	01	02
02	52	02	04
03	54	03	08
04	57	04	10
05	58	05	20
06	5B	06	40
07	5D		
08	23		
09	25		
10	26		
11	29		
12	2A		
13	2C		
14	31		
15	34		
16	0E		
17	0B		
18	13		
19	15		
20	16		
21	19		

Table 3–26 (Cont.): Syndromes for Single-Bit Errors

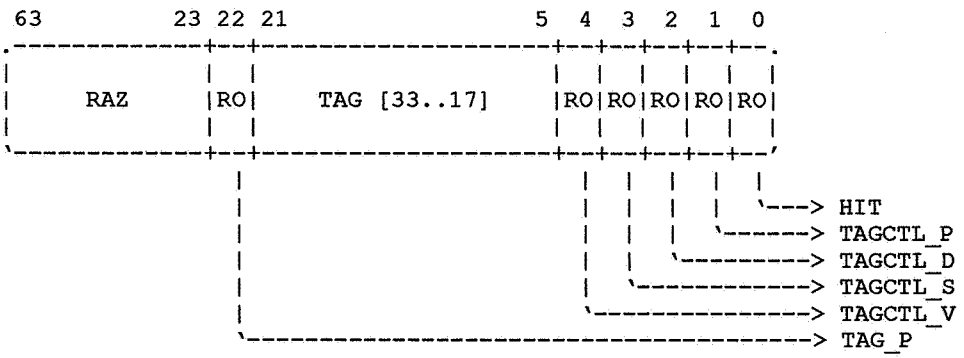
Data Bit	Syndrome(Hex)	Check Bit	Syndrome(Hex)
22	1A		
23	1C		
24	62		
25	64		
26	67		
27	68		
28	6B		
29	6D		
30	70		
31	75		

3.10.7 BC_TAG

BC_TAG is a read-only IPR. Unless locked, the BC_TAG register is loaded with the results of every backup cache tag probe. When a tag or tag control parity error or primary fill data error (parity or ECC) occurs this register is locked against further updates. Software may read the LSB of this register by using the HW_MFPR instruction. Each time an HW_MFPR from BC_TAG completes the contents of BC_TAG are shifted one bit position to the right, so that the entire register may be read using a sequence of HW_MFPRs. Software may unlock the BC_TAG register using a HW_MTPR to BC_TAG.

Successive HW_MFPRs from the BC_TAG register must be separated by at least one null cycle.

Figure 3–31: Backup Cache Tag Register (BC_TAG)



Unused tag bits in the TAG field of this register are always clear, based on the size of the external cache as determined by the BC_SIZE field of the BIU_CTL register.

3.11 ECC Error Correction

When in ECC mode, the 21064 generates longword ECC on writes, and checks ECC on reads. The 21064 does not include hardware to correct single-bit errors, however.

When an ECC error is recognized during a Dcache fill the BIU places the affected fill block into the Dcache unchanged, validates the block and posts a machine check. The load instruction which triggered the Dcache fill is completed by writing the requested longword(s) into the register file. The longword(s) read by the load instruction may or not have been the cause of the error, but a machine check is posted either way. The Ibox will react to the machine check by aborting instruction execution before any instruction issued subsequent to the load could overwrite the register containing the load data, and vectoring to the PAL code machine check handler. Sufficient state is retained in various status registers (see Section 3.12) for PAL code to determine whether the error affects the longword(s) read by the load instruction, and whether the error is correctable. In any event, PAL code must explicitly flush the Dcache. If the longword containing the error was written into the register file, PAL code must either correct it and restart the machine, or report an uncorrectable hardware error to the operating system. Independent of whether the failing longword was read by the load instruction, PAL may scrub memory by explicitly reading the longword with the physical/lock variant of the HW_LD instruction, flipping the necessary bit, and writing the longword with the physical/conditional variant of the HW_ST instruction. Note that when PAL rereads the affected longword the hardware may report no errors, indicating that the longword has been overwritten.

When an ECC error occurs during an Icache fill the BIU places the affected fill block into the Icache unchanged, validates the block and posts a machine check. The Ibox will vector to the PAL code machine check handler before it executes any of the instructions in the bad block. PAL code may then flush the Icache and scrub memory as described above.

As compared with hardware error correction, this approach is vulnerable to single-bit errors which may occur during I-stream reads of the PAL code machine check handler, to single-bit errors which occur in multiple quadwords of a cache fill block, and to single-bit errors which occur as a result of multiple silo'ed load misses.

3.12 Error Flows

The following sections give a summary of the hardware flows for various error conditions.

3.12.1 I-stream ECC error

- data put into Icache unchanged, block gets validated
- machine check
- BIU_STAT: FILL_ECC, FILL_IRD set, FILL_SEO set if multiple errors occurred
- FILL_ADDR[33..5] & BIU_STAT[FILL_QW] give bad QW's address
- FILL_SYNDROME contains syndrome bits associated with failing quadword
- BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
- DC_STAT locked - contents are UNPREDICTABLE
- BC_TAG holds results of external cache tag probe if external cache was enabled for this transaction

3.12.2 D-stream ECC error

- data put into Dcache unchanged, block gets validated
- machine check
- BIU_STAT: FILL_ECC set, FILL_IRD clear, FILL_SEO set if multiple errors occurred
- FILL_ADDR[33..5] & BIU_STAT[FILL_QW] give bad QW's address
- FILL_ADDR[4..2] contain PA bits [4..2] of location which the failing load instruction attempted to read
- FILL_SYNDROME contains syndrome bits associated with failing quadword
- BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
- DC_STAT: RA identifies register which holds the bad data. LW,LOCK,INT,VAX_FP identify type of load instruction
- BC_TAG holds results of external cache tag probe if external cache was enabled for this transaction

3.12.3 BIU: tag address parity error

- recognized at end of tag probe sequence
- lookup uses predicted parity so transaction misses the external cache
- BC_TAG holds results of external cache tag probe
- machine check
- BIU_STAT: BC_TPERR set
- BIU_ADDR holds address

3.12.4 BIU: tag control parity error

- recognized at end of tag probe sequence
- transaction forced to miss external cache
- BC_TAG holds results of external cache tag probe
- machine check
- BIU_STAT: BC_TCPERR set
- BIU_ADDR holds address

3.12.5 BIU: system external transaction terminated with CACK_SERR

- CRD interrupt.
- BIU_STAT: BIU_SERR set, BIU_CMD holds cReq_h[2..0].
- BIU_ADDR holds address.

3.12.6 BIU: system transaction terminated with CACK_HERR

- machine check
- BIU_STAT: BIU_HERR set, BIU_CMD holds cReq_h[2..0]
- BIU_ADDR holds address

3.12.7 BIU: I-stream parity error (parity mode only)

- data put into Icache unchanged, block gets validated
- machine check
- BIU_STAT: FILL_DPERR set, FILL_IRD set
- FILL_ADDR[33..5] & BIU_STAT[FILL_QW] give bad QW's address
- FILL_SYNDROME identifies failing longword(s)
- BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
- DC_STAT locked - contents are UNPREDICTABLE

- BC_TAG holds results of external cache tag probe if external cache was enabled for this transaction

3.12.8 BIU: D-stream parity error (parity mode only)

- data put into Dcache unchanged, block gets validated
- machine check
- BIU_STAT: FILL_DPERR set, FILL_IRD clear
- FILL_ADDR[33..5] & BIU_STAT[FILL_QW] give bad QW's address
- FILL_ADDR[4..2] contain PA bits [4..2] of location which the failing load instruction attempted to read
- FILL_SYNDROME identifies failing longword(s)
- BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
- DC_STAT: RA identifies register which holds the bad data. LW,LOCK,INT,VAX_FP identify type of load instruction
- BC_TAG holds results of external cache tag probe if external cache was enabled for this transaction

Chapter 4
External Interface

4.1 Overview

The 21064 chip connects directly to an external cache built from off-the-shelf static RAMs. Because building high-speed logic is very difficult in low-end systems, the chip controls the RAMs directly. The chip contains a programmable external cache interface, so that each system can make its own external cache speed and configuration tradeoffs.

The clocks used by the external interface are generated by the chip, but the speed of the clocks is programmable, and is determined during chip reset. This allows each system to make its own external interface speed tradeoffs. The 21064 is configured during reset to use either a 64-bit or 128-bit wide external data bus. The bulk of this chapter describes the chip's operation in 128-bit mode, and Section 4.3 of this chapter describes details specific to 64-bit mode operation.

4.2 Signals

The following table lists all of the signals on the chip. In the "type" column, an "I" means a pin is an input, an "O" means the pin is an output, and a "B" means the pin is bidirectional.

Table 4–1: 21064 Signal Pins

Signal Name	Count	Type	Function
clkIn_h, _l	2	I	Clock input
testClkIn_h, _l	2	I	Clock input for testing
cpuClkOut_h	1	O	CPU clock output
sysClkOut1_h, _l	2	O	System clock output, normal
sysClkOut2_h, _l	2	O	System clock output, delayed
dcOk_h	1	I	Power and clocks ok
reset_l	1	I	Reset
icMode_h 1..0	2	I	Icache Test Mode Selection
sRomOE_l	1	O	Serial ROM output enable
sRomD_h	1	I	Serial ROM data/Rx data
sRomClk_h	1	O	Serial ROM clock/Tx data
adr_h 33..5	29	B	Address bus
data_h 127..0	128	B	Data bus
check_h 27..0	28	B	Check bit bus
dOE_l	1	I	Data bus output enable
dWSel_h 1..0	2	I	Data bus write data select
dRAck_h 2..0	3	I	Data bus read data acknowledge
tagCEOE_h	1	O	tagCtl and tagAdr CE/OE
tagCtlWE_h	1	O	tagCtl WE
tagCtlV_h	1	B	Tag valid
tagCtlS_h	1	B	Tag shared
tagCtlD_h	1	B	Tag dirty
tagCtlP_h	1	B	Tag V/S/D parity
tagAdr_h 33..17	17	I	Tag address
tagAdrP_h	1	I	Tag address parity (Cont.)
tagOk_h, _l	2	I	Tag access from CPU is ok

Table 4–1 (Cont.): 21064 Signal Pins

Signal Name	Count	Type	Function
tagEq_l	1	O	Tag compare output
dataCEOE_h 3..0	4	O	data CE/OE, longword
dataWE_h 3..0	4	O	data WE, longword
dataA_h 4..3	2	O	data A[4..3]
holdReq_h	1	I	Hold request
holdAck_h	1	O	Hold acknowledge
cReq_h 2..0	3	O	Cycle request
cWMask_h 7..0	8	O	Cycle write mask
cAck_h 2..0	3	I	Cycle acknowledge
iAdr_h 12..5	8	I	Invalidate address
dInvReq_h	1	I	Invalidate request, Dcache
dMapWE_h	1	O	Backmap WE, Dcache
irq_h 5..0	6	I	Interrupt requests
vRef	1	I	Input reference
eclOut_h	1	I	Output mode selection
perf_cnt_h 1..0	2	I	Performance counter inputs
tristate_l	1	I	Set pins to high impedance state for testing
cont_l	1	I	Continuity for testing

Systems using 21064 in 128-bit mode should ignore dataA_h 3 and tie dWSel_h 0 false. See Section 4.3 for 64-bit mode details.

4.2.1 Clocks

External logic supplies the 21064 with a differential clock at twice the desired internal clock frequency via the clkIn_h and clkIn_l pins. This clock is divided by 2 to generate the internal chip clock.

The internal chip clock is supplied to the external interface via the cpuClkOut_h pin. The false-to-true transition of cpuClkOut_h is the "CPU clock" used in the timing specification for the tagOk_h,_l signals.

The CPU clock is divided by a programmable value between 2 and 8 to generate a system clock, which is supplied to the external interface via the sysClkOut1_h and sysClkOut1_l pins. The system clock is delayed by a programmable number of CPU clock cycles between 0 and 3 to generate a delayed system clock, which is supplied to the external interface via the sysClkOut2_h and sysClkOut2_l pins.

The clock generator runs generating cpuClkOut_h and correctly timed and positioned sysClkOut1 and sysClkOut2, while the chip is held in reset.

The output of the programmable divider is symmetric if the divisor is even and asymmetric with sysClkOut1_h TRUE for one extra CPU cycle if the divisor is odd.

The false-to-true transition of sysClkOut1_h is the "system clock" used as a timing reference throughout this specification.

Almost all transactions on the external interface run synchronously to the CPU clock and phase aligned to the system clock. The external interface appears to be running synchronously to the system clock (most setup and hold times are referenced to the system clock). The exceptions to this are the fast CPU controlled transactions on the external caches and the sample of the tagOk_h, _l inputs, which are synchronous to the CPU clock, but independent of the system clock.

4.2.2 DC_OK and Reset

The 21064 contains a ring oscillator which is switched into service during power up to provide an internal chip clock. The dcOk_h signal switches clock sources between an on-chip ring oscillator and the external clock oscillator. If dcOk_h is false then the on-chip ring oscillator feeds the clock generator and the chip is held in reset independent of the state of the reset_l signal. If dcOk_h is true, then the external clock oscillator feeds the clock generator. When dcOk_h is true the vRef input must be valid so that inputs can be sensed. The dcOk_h signal is special in that it does not require that vRef be stable to be sensed. It is important to drive dcOk_h false until the voltage on vRef has stabilized. Because chip testers can apply clocks and power to the chip at the same time, the chip tester can always drive dcOk_h true, but the tester must drive reset_l true for a period longer than the minimum hold time of vRef.

When the 21064 is running off the internal ring oscillator the clock outputs follow it, just like they would when real clocks are applied. The frequency of the ring oscillator varies from chip to chip within a range of 10MHZ to 100MHz, which corresponds to an internal CPU clock frequency of between 5 MHz and 50 MHz. Also, when the dcOk_h signal is false, the system clock divisor is forced to eight, and the sysClkOut2_h, _l delay is forced to three.

If the dcOk_h signal is generated by an RC delay, there is no check to verify that the input clocks are really running. If a board is powered up in manufacturing with a missing, defective, or mis-soldered clock oscillator, then the 21064 will enter a possibly destructive high-current state. Furthermore, if a clock oscillator fails in stage 1 burn-in then the 21064 can also enter this state. The frequency and duration of such events need to be understood by the module designer to decide if this is really a problem.

The reset_l signal forces the CPU into a known state - see Table 3–8. The reset_l signal can be asynchronous, and need not be asserted beyond the assertion of dcOk_h to guarantee that the chip is properly reset.

In order to bring the chip out of internal reset at a deterministic time, the reset_l pin can be deasserted synchronously with respect to the system clock. See chapter Chapter 6 for the setup and hold requirements of the reset_l pin when used in this way.

The 21064 CPU chip uses a 3.3V power supply. This 3.3V supply must be stable before any input goes above 4V.

While in reset, sysClkOut and external bus configuration information is read from the irq_h pins. External logic should drive the configuration information onto the irq_h pins any time reset_l is true.

The irq_h 5 bit is used to select 128-bit or 64-bit mode. If irq_h 5 is true then 128-bit mode is selected.

The irq_h 2..0 bits encode the value of the divisor used to generate the system clock from the CPU clock.

Table 4–2: System Clock Divisor

irq_h 2	irq_h 1	irq_h 0	Ratio
F	F	F	2
F	F	T	3
F	T	F	4
F	T	T	5
T	F	F	6
T	F	T	7
T	T	F	8
T	T	T	8

The irq_h 4..3 bits encode the delay, in CPU clock cycles, from sysClkOut1 to sysClkOut2.

Table 4–3: System Clock Delay

irq_h 4	irq_h 3	Delay
irq_h 4	irq_h 3	Delay
F	F	0
F	T	1
T	F	2
T	T	3

When the tristate_l pin is asserted the chip is internally forced into the reset state, without resampling the interrupt pins.

4.2.3 Initialization and Diagnostic Interface

The 21064 implements three Icache initialization modes to support the chip and module level testing. The value placed on icMode_h 1..0 determines which of these modes is used after the 21064 is reset. Unlike the value placed on irq_h 5..0 during reset, the value placed on icMode_h 1..0 must be retained after reset_l is deasserted.

Table 4–4: Icache Test Modes

icMode_h 1	icMode_h 0	Mode
F	F	Serial Rom
F	T	Disabled
T	F	Icache Test - Write
T	T	Icache Test - Read

If the value on icMode_h 1..0 selects Serial ROM Mode, the 21064 will load the contents of its internal Icache from an external serial ROM (such as an AMD Am1736) before executing its first instruction. The serial ROM could contain as many instructions as needed to complete the configuration of the external interface, e.g. setting the timing on the external cache RAMs and diagnose the path between the CPU chip and the real ROM. The 21064 is in PALmode following the deassertion of reset_l. This gives the code loaded into the Icache access to all of the visible state within the chip.

Three signals are used to interface to the serial ROM. The sRomOE_l output signal supplies the output enable to the ROM, serving both as an output enable and as a reset (refer to the serial ROM specifications for details). The sRomClk_h output signal supplies the clock to the ROM that causes it to advance to the next bit. The ROM data is read via the sRomD_h input signal.

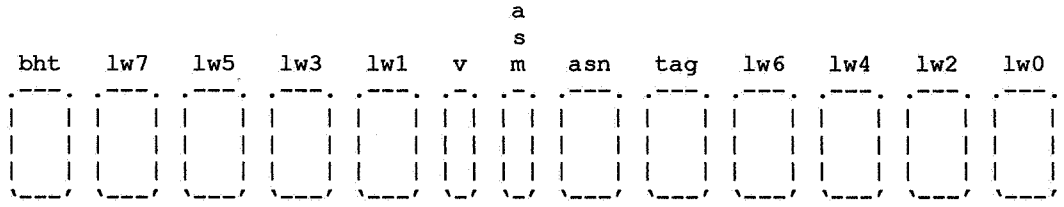
Once the data in the serial ROM has been loaded into the Icache, the three special signals become simple parallel I/O pins that can be used to drive a diagnostic terminal. When the serial ROM is not being read, the sRomOE_l output signal is false. If this pin is wired to the active high enable of an RS422 receiver driving onto sRomD_h (the 26LS32 will work) and to the active high enable of an RS422 driver driving from sRomClk_h (the 26LS31 will work). The CPU allows sRomD_h to be read and sRomClk_h to be written by PALcode; this is sufficient hardware support to implement a bit-banged serial interface.

Using the icMode_h 1..0 pins, the Icache diagnostic interface may be disabled altogether. In this case, since the Icache valid bits are cleared by reset, the first instruction fetch will miss the Icache.

In addition to Serial ROM mode, the 21064 includes two test modes which together allow chip tester hardware full read and write access to the Icache. Icache Test/Write Mode works exactly like Serial ROM mode except that bits are loaded into the Icache at a higher rate. Icache Test/Read Mode allows the contents of the Icache to be read in a bit-serial manner from the sRomOE_l pin. These two modes are available only to chip test hardware. Systems using the 21064 must tie icMode_h 1 to FALSE.

In the 21064, all Icache bits are loaded from the diagnostic interface, including each blocks' tag, ASN, ASM, valid and branch history bits. The Icache blocks are loaded in sequential order starting with block zero and ending with block 255. The order in which bits within each block are serially loaded is shown below:

Figure 4–1: Serial RAM Load - Block Ordering



Bits within each field are arranged such that high-order bits are on the left. The serial chain shifts to the right.

When the Icache is loaded the valid bit in each cache block is set, and the tag is cleared. Conceptually, the data bits from the serial ROM are shifted into a 64-bit wide holding register and then written into the Icache 64 bits at a time. The bits from the serial ROM are shifted into this holding register from the least significant bit to the most significant bit. Quadwords are written into the Icache in increasing order starting with the quadword at byte address zero.

4.2.4 Address Bus

The three state, bidirectional adr_h pins provide a path for addresses to flow between the 21064 and the rest of the system. The adr_h pins are connected to the buffers that drive the address pins of the external cache RAMs and to the transceivers that are located between the 21064 local address bus and the CPU module address bus.

The address bus is normally driven by the processor. The 21064 stops driving the address bus during reset and during external cache hold. In the external cache hold state the address bus acts like an input, and the tagEq_l output is the result of an equality compare between adr_h and tagAdr_h. Only bits that are part of the cache tag, as specified by the BC_SIZE field of the BIU_CTL IPR, participate in the compare. The tagEq_l pin is asserted during external cache hold only if the result of the tag comparison is true, and the parity calculated across the appropriate bits of tagAdr_h matches the value on tagAdrP_h. Even parity is used. tagEq_l is deasserted when the address bus is not in the external cache hold state.

4.2.5 Data Bus

The three state, bidirectional data_h pins provide a path for data to flow between the 21064 and the rest of the system. The data_h pins connect directly to the I/O pins of the external cache data RAMs and to the transceivers that are located between the 21064 local data bus and the CPU module data bus.

The three state, bidirectional check_h pins provide a path for check bits to flow between the CPU and the rest of the system. The check_h pins connect directly to the I/O pins of the external cache data RAMs, to the transceivers that are located between the 21064 local check bus, and the CPU module check bus.

The data bus is driven by the 21064 when it is running a fast write cycle on the external caches or when some type of write cycle has been presented to the external interface and external logic has enabled the data bus drivers (via dOE_l).

If the 21064 is in ECC mode then the check_h pins carry 7 check bits for each longword on the data bus. Bits check_h 6..0 are the check bits for data_h 31..0. Bits check_h 13..7 are the check bits for data_h 63..32. Bits check_h 20..14 are the check bits for data_h 95..64. Bits check_h 17..21 are the check bits for data_h 127..96.

The following ECC code is used. This code is the same one used by the IDT49C460 and AMD29C660 32-bit ECC generator/checker chips.

Figure 4–2: ECC Code

```
          ddddddddddddddddddddddddddddddd
          33222222222211111111110000000000
          10987654321098765432109876543210
c6 XOR    xxxxxxxx                 xxxxxxxx
c5 XOR    xxxxxxxx         xxxxxxxx
c4 XOR    xx       xxxxxx  xx       xxxxxx
c3 XNOR     xxx    xxx    xx  xxx    xxx    xx
c2 XNOR x x  xx x  xx  xx x  xx x  xx  x
c1 XOR       x x x x x xxx    x x x x x x xxx
c0 XOR    x xx x     x xxx  x  x xxxx x    x
```

By arranging the data and check bits correctly, it is possible to arrange that any number of errors restricted to a 4-bit group can be detected. One such arrangement is as follows:

Figure 4–3: Example of Errors Detected

```
d 00,  d 01,  d 03,  d 25
d 02,  d 04,  d 06,  c 06
d 05,  d 07,  d 12,  c 03
d 08,  d 09,  d 11,  d 14
d 10,  d 13,  d 15,  d 19
d 16,  d 17,  d 22,  d 28
d 18,  d 23,  d 30,  c 05
d 20,  d 27,  c 04,  c 00
d 21,  d 26,  c 02,  c 01
d 24,  d 29,  d 31
```

If the 21064 is in PARITY mode, then 4 of the check_h pins carry EVEN parity for each longword on the data bus and the rest of the bits are unused. Bit check_h 0 is the parity bit for data_h 31..0. Bit check_h 7 is the parity bit for data_h 63..32. Bit check_h 14 is the parity bit for data_h 95..64. Bit check_h 21 is the parity bit for data_h 127..96.

The ECC bit in the BIU_CTL IPR determines if the 21064 is in ECC mode or in PARITY mode.

4.2.6 External Cache Control

The external cache is a direct-mapped, write-back cache. The 21064 always views the external cache as having a tag for each 32-byte block (the same as the on-chip Icache and Dcache).

The external cache tag RAMs are located between the 21064's local address bus and the 21064's tag inputs. The external cache data RAMs are located between the CPU's local address bus and the CPU's local data bus. The 21064 reads the external cache tag RAMs to determine if it can complete a cycle without any module level action. The 21064 reads or writes the external cache data RAMs, if this is the case.

A cycle requires no module level action if it is a non-LDxL read hit to a valid block, or a non-STxC write hit to a valid but not shared block. All other cycles require module level action. All cycles require module level action if the external cache is disabled (the BC_EN bit in the BIU_CTL IPR is cleared) or the physical address of the reference is in a quadrant in memory that is not cached, i.e. the appropriate bit in the BC_PA_DIS field in the BIU_CTL IPR is set for the quadrant of the reference.

All 21064 controlled cycles on the external cache have fixed timing, described in terms of the 21064's internal clock. The actual timing of the cycle is programmable (via the BC_RD_SPD, BC_ WR_SPD, and BC_WE_CTL fields in the BIU_CTL IPR), allowing for much flexibility in the choice of CPU clock frequencies and cache RAM speeds.

The external cache RAMs can be partitioned into three sections; the tagAdr RAM, the tagCtl RAM, and the data RAM. Sections do not straddle physical RAM chips.

4.2.6.1 The TagAdr RAM

The tagAdr RAM contains the high order address bits associated with the external cache block, along with a parity bit. The contents of the tagAdr RAM is fed to the on-chip address comparator and parity checker via the tagAdr_h and tagAdrP_h inputs.

The 21064 verifies that tagAdrP_h is an EVEN parity bit over tagAdr_h when it reads the tagAdr RAM. If the parity is wrong, the tag probe results in a miss and an external transaction is initiated. If machine checks are enabled, (the MCHK_EN bit in the Abox_CTL IPR is set) the 21064 traps to PALcode.

The number of bits of tagAdr_h that participate in the address compare and the parity check is controlled by the BC_SIZE field in the BIU_CTL IPR. The tagAdr_h signals go all the way down to address bit 17, allowing for a 128 Kbyte cache built out of RAMs that are 8K deep.

The chip enable or output enable for the tagAdr RAM is normally driven by a two input NOR gate (such as the 74AS805B). One input of the NOR gate is driven by tagCEOE_h and the other input is driven by external logic. The 21064 drives tagCEOE_h false during reset, during external cache hold, and during any external cycle. The OE bit in the BIU_CTL IPR determines if tagCEOE_h has chip enable timing or output enable timing.

4.2.6.2 The TagCtl RAM

The tagCtl RAM contains control bits associated with the external cache block, along with a parity bit. The 21064 reads the tagCtl RAM via the three tagCtl signals to determine the state of the block. The 21064 writes the tagCtl RAM via the three tagCtl signals to make blocks dirty.

The 21064 verifies that tagCtlP_h is an EVEN parity bit over tagCtlV_h, tagCtlS_h, and tagCtlD_h when it reads the tagCtl RAM. If the parity is wrong, the tag probe results in a miss, and an external transaction is initiated. If machine checks are enabled (the MCHK_EN bit in the Abox_CTL IPR is set), the 21064 traps to PALcode. The 21064 computes EVEN parity across the tagCtlV_ h, tagCtlS_h, and tagCtlD_h bits and drives the result onto the tagCtlP_h pin, when it writes the tagCtl RAM.

The following combinations of the tagCtl RAM bits are allowed. Note that the bias toward conditional write-through coherence is really only in name. The tagCtlS_h bit can be viewed simply as a write protect bit.

Table 4–5: Tag Control Encodings

tagCtlV_h	tagCtlS_h	tagCtlD_h	Meaning
F	X	X	Invalid
T	F	F	Valid, private
T	F	T	Valid, private, dirty
T	T	F	Valid, shared
T	T	T	Valid, shared, dirty

The 21064 can satisfy a read probe if the tagCtl bits indicate the entry is valid (tagCtlV_h = T). The 21064 can satisfy a write probe if the tagCtl bits indicate the entry is valid and not shared (tagCtlV_h = T, tagCtlS_h = F).

The chip enable or output enable for the tagCtl RAM is normally driven by a two input NOR gate (such as the 74AS805B). One input of the NOR gate is driven by tagCEOE_h and the other input is driven by external logic. The 21064 drives tagCEOE_h false during reset, external cache hold, and any external cycle. The OE bit in the BIU_CTL IPR determines if tagCEOE_h has chip enable timing or output enable timing.

The write enable for the tagCtl RAM is normally driven by a two input NOR gate. One input of the NOR gate is driven by tagCtlWE_h and the other input is driven by external logic. The 21064 drives tagCtlWE_h false during reset, during external cache hold, and during any external cycle. The BC_WE_CTL field in the BIU_CTL IPR determines the width of the write enable and its position within the write cycle.

4.2.6.3 The Data RAM

The data RAM contains the actual cache data along with any ECC or parity bits.

The most significant bits of the data RAM address are driven, via buffers, from the address bus. The least significant bit of the data RAM address is driven by a two input NOR gate (such as the 74AS805B). One of the inputs of the NOR gate is driven by dataA_ h 4 and the other input is driven by external logic. The 21064 drives dataA_h 4 false during reset, external cache hold, and any external cycle.

The chip enables or output enables for the data RAM are driven by a two input NOR gate. One input of the NOR gate is driven by dataCEOE_h 3..0 and the other input is driven by external logic. The 21064 drives dataCEOE_h 3..0 false during reset, external cache hold, and external cycles. The OE bit in the BIU_CTL IPR determines if dataCEOE_h 3..0 has chip enable timing or output enable timing.

The write enables for the data RAM are normally driven by a two input NOR gate (such as the 74AS805B). One input of the NOR gate is driven by dataWE_h 3..0 and the other input is driven by external logic. The 21064 drives dataWE_h 3..0 false during reset, external cache hold, and any external cycle. The BC_WE_CTL field in the BIU_CTL IPR determines the width of the write enable and its position within the write cycle.

4.2.6.4 Backmap

Some systems may wish to maintain a backmap of the contents of the primary data cache to improve the quality of their invalidate filtering. The 21064 must maintain the backmap for external cache read hits, since external cache read hits are controlled totally by the 21064. External logic maintains the backmaps for external cycles (read misses, invalidates, and so on).

The backmap is only consulted by external logic. Its format and existence is of no concern to the 21064. All the 21064 does is generate a backmap write pulse at the right time. Simple systems will not bother to maintain a backmap, will not connect the backmap write pulse to anything, and will generate extra invalidates.

The write enable input of the data cache backmap RAM is driven by a two input NOR gate (such as the 74AS805B). One side of the NOR gate is driven by dMapWE_h and the other input is driven by external logic. The CPU drives a write pulse onto dMapWE_h whenever is fills the on-chip data cache from the external cache.

In 128-bit mode the dMapWE_h 1..0 signals assert one CPU cycle into the second (last) data read cycle and negate one CPU cycle from the end of that cycle. If read cycles are 3 CPU cycles long, then dMapWE_h is one CPU cycle long. See Section 4.3 for 64-bit mode operations.

Note

This is normally caused by the fact that the backmap write overlaps a cycle whose length is specified by BC_RD_SPD. If we used the standard write pulse timing mechanism and BC_WR_SPD were longer than BC_RD_SPD, the address would go away in the middle of the write cycle.

4.2.6.5 External Cache Access

The external caches are normally controlled by the 21064. Two methods exist for gaining access to the external cache RAMs.

4.2.6.5.1 HoldReq and HoldAck

The simple method for external logic to access the external caches is to assert the holdReq_h signal. When holdReq_h is asserted, the 21064 finishes any external cache cycle which may be in progress, three states adr_h, data_h, check_h, tagCtlV_h, tagCtlD_h, tagCtlS_h, and tagCtlP_h drives tagCEOE_h, tagCtlWE_h, dataCEOE_h, data_WE_h, and dataA_h false and asserts holdAck_h. The cReq_h and cWMask_h signals are not modified in any way. When external logic is finished with the external caches it deasserts holdReq_h. When the 21064 detects the deassertion of holdReq_h it deasserts holdAck_h and re-enables its outputs.

The holdReq_h signal is synchronous. External logic must guarantee setup and hold requirements with respect to the system clock. The holdAck_h signal is synchronous to the CPU clock but phase aligned to the system clock, so it can be used as an input to state machines running off the system clock.

The 21064 generates the holdAck_h signal such that it can be tied directly to the enable-inputs of external three state drivers which connect to the bidirectional pin bus signals. The 21064 will turn off its three state drivers on or before the system clock edge, at which time it asserts holdAck_h and will turn on its three state drivers two CPU cycles after the system clock edge at which it deasserts holdAck_h.

The delay from holdReq_h assertion to holdAck_h assertion depends on the programming of the external interface and on exactly how the system clock is aligned with a pending external cache cycle. In the best case the external cache is idle or is just about to start a cycle. In which case, holdAck_h asserts one system clock cycle after the system clock edge, at which time 21064 samples the holdReq_h assertion. In the worst case at the system clock edge at which 21064 samples the holdReq_h assertion happens one CPU clock cycle into an external cache write probe that hits on a non shared line and requires two RAM data cycles to complete. In this case, holdAck_h asserts at the first system clock edge that is at least ((BC_RD_SPD + 1) - 1) + 2*(BC_WR_SPD + 1) + 1 CPU cycles after the system clock edge at which time the 21064 sampled the holdReq_h assertion.

HoldAck_h deasserts in the system clock cycle immediately following the system clock edge at which time the deassertion of holdReq_h is sampled.

A holdReq_h/holdAck_h sequence can happen at any time, even in the middle of an external transaction. In this case all of the acknowledge-like signals (dOE_l dWSel_h, dRAck_h, cAck_h) work normally. Although the 21064 has forced most of its outputs to either three state or false. Doing anything useful with them is difficult.

The assertion of holdReq_h prevents the BIU sequencer from starting new CPU requests. However, if the BIU sequencer has already started an external cache tag probe when holdReq_h is asserted and the result of the tag probe is such that an external transaction is required to complete the CPU's request. The BIU sequencer will initiate the external transaction by driving the cReq_h signals to the appropriate value despite holdReq_h's assertion. The holdAck_h signal will assert at the next system clock edge after the tag probe completes.

The 21064 does not turn on its three state drivers until two CPU cycles after it deasserts holdAck_h. Care must be taken as to when external logic begins processing new external transactions at the tail end of a holdReq_h/holdAck_h sequence.

4.2.6.5.2 TagOk

The fastest way for external logic to gain access to the external caches is to use the tagOk_ h, _l signals. The TagOk_h, _l signals are 21064 bus interface control signals which allow external logic to stall a CPU cycle on the external cache RAMs at the last possible instant. All tradeoffs surrounding these signals have been made in favor of high-performance systems, making them next to impossible to use in low-end systems.

The tagOk_h and tagOk_l signals are synchronous. External logic must guarantee setup and hold requirements with respect to the CPU clock. This implies very fast logic, since the CPU clock may run at 200 MHz for the binned parts.

The only thing that tagOk does is stall a sequencer in the 21064 bus interface unit. The 21064 does not tri-state the busses that run between the CPU and the external cache RAMs. External logic must supply the necessary multiplexing functions in the address and data path.

If the tagOk is true at a CPU clock edge, the external logic is guaranteeing that the tagCtl and tagAdr RAMs were owned by the 21064 in the previous BC_RD_SPD+1 CPU cycles, that the tagCtl RAMs will be owned by the 21064 in the next BC_WR_SPD+1 cycles, that the data RAMs were owned by the 21064 in the previous BC_RD_SPD+1 cycles, and that the data RAMs will be owned by the 21064 in the next BC_RD_SPD+1 CPU cycles or in the next 2*(BC_WR_SPD+1) CPU cycles, whichever is longer.

The bus interface unit samples tagOk in the last two cycles of each tag probe and only proceeds if tagOk was asserted in both of these cycles. Two cycles of tagOk assertion rather then one was chosen to alleviate a tight circuit path inside the chip. This choice in no way impacts the above stated use of tagOk by external logic. If the 21064 samples tagOk as false in either of the last two CPU cycles of a tag probe, then it stalls until it samples tagOk true in consecutive cycles (at which time all of the above assertions are true, which means that any address the 21064 has been holding on the address bus has made it through the external cache RAMs) and then it proceeds normally.

4.2.7 External Cycle Control

On READ_BLOCK and LDxL cycles, the cWMask_h pins have additional information about the cache miss overloaded onto them. The cWMask_h 1:0 and cWMask_h 3 pins contain miss address bits 4:3 and 2 respectively. These additional address bits, which specify the longword that missed are needed to implement longword granularity to I/O devices.

An external cycle begins when the 21064 puts a cycle type onto the cReq_h outputs. Some cycles put an address on the adr_h outputs and additional information (low-order address bits, I/D stream indication, write masks) on the cWMask_h outputs. All of these outputs are synchronous and the 21064 meets setup and hold requirements with respect to the system clock.

The cycle types are as follows.

Table 4–6: Cycle Types

cReq_h 2	cReq_h 1	cReq_h 0	Type
F	F	F	IDLE
F	F	T	BARRIER
F	T	F	FETCH
F	T	T	FETCHM
T	F	F	READ_BLOCK
T	F	T	WRITE_BLOCK
T	T	F	LDxL
T	T	T	STxC

A BARRIER cycle is generated by the MB instruction. Normally all the module does with this cycle is acknowledge it. Modules which have write buffers between the 21064 and the memory system must drain these buffers before the cycle is acknowledged. This guarantees that machine checks caused by transport and/or memory system errors get posted on the correct side of the MB instruction.

The FETCH and FETCHM cycles are generated by the FETCH and FETCHM instructions, respectively. The address bus contains the effective address of the FETCH or FETCHM instruction. These addresses can be used by module level prefetching logic. Simple systems simply acknowledge the cycles.

The READ_BLOCK cycle is generated on read misses. External logic reads the addressed block from memory and supplies it (128 bits at a time) to the 21064 via the data bus. External logic may also write the data into the external cache, after perhaps writing a victim.

The WRITE_BLOCK cycle is generated on write misses and on writes to shared blocks. External logic pulls the write data (128 bits at a time) from the 21064 via the data bus and writes the valid longwords to memory. External logic may also write the data into the external cache, after perhaps writing a victim.

The LDxL cycle is generated by the interlocked load instructions. The cycle works just like a READ_BLOCK, although the external cache has not been probed (so the external logic needs to check for hits) and the address has to be latched into a locked address register.

The STxC cycle is generated by the conditional store instructions. The cycle works just like a WRITE_BLOCK, although the external cache has not been probed (so that external logic needs to check for hits) and the cycle can be acknowledged with a failure status.

On WRITE_BLOCK and STxC cycles the cWMask_h pins supply longword write masks to the external logic, indicating which longwords in the 32-byte block are valid. A cWMask_ h bit is true if the longword is valid. WRITE_BLOCK commands can have any combination of mask bits set. STxC cycles can only have combinations that correspond to a single longword or quadword.

On READ_BLOCK and LDxL cycles the cWMask_h pins have additional information about the miss overloaded onto them. The cWMask_h 1..0 pins contain miss address bits 4..3 (indicating the address of the quadword that actually missed), which is needed to implement quadword read granularity to I/O devices. The cWMask_h 2 pin is true if the miss is a D-stream reference and false if the miss is an I-stream reference.

The cycle remains on the external interface until external logic acknowledges it, by placing an acknowledgment type on the cAck_h pins. The cAck_h inputs are synchronous, and external logic must guarantee setup and hold requirements with respect to the system clock.

The acknowledgment types are as follows.

Table 4–7: Acknowledgment Types

cAck_h 2	cAck_h 1	cAck_h 0	Type
F	F	F	IDLE
F	F	T	HARD_ERROR
F	T	F	SOFT_ERROR
F	T	T	STxC_FAIL
T	F	F	OK

The 21064 behavior in response to cAck_h encodings others than those listed above is UNDEFINED.

The HARD_ERROR type indicates that the cycle has failed in some catastrophic manner. The 21064 latches sufficient state to determine the cause of the error and initiates a machine check.

The SOFT_ERROR type indicates that a failure occurred during the cycle, but the failure was corrected. The 21064 latches sufficient state to determine the cause of the error and initiates a corrected error interrupt.

The STxC_FAIL type indicates that a STxC cycle has failed. It is UNDEFINED what happens if this type is used on anything but an STxC cycle.

The OK type indicates success.

The dRAck_h pins inform the 21064 that read data is valid on the data bus, if the data should be cached, and if ECC checking and correction or parity checking should be attempted. The dRAck_h inputs are synchronous. External logic must guarantee setup and hold requirements with respect to the system clock. If dRAck_h is sampled IDLE at a system clock, then the data bus is ignored. If dRAck_h is sampled non IDLE at a system clock, then the data bus is latched at that system clock and external logic must guarantee that the data meets setup and hold with respect to the system clock.

The acknowledgment types are as follows.

Table 4–8: Read Data Acknowledgment Types

dRAck_h 2	dRAck_h1	dRAck_h 0	Type
F	F	F	IDLE
T	F	F	OK_NCACHE_NCHK
T	F	T	OK_NCACHE
T	T	F	OK_NCHK
T	T	T	OK

The 21064 behavior in response to dRAck_h encoding others than those listed above is UNDEFINED.

The first non IDLE sample of dRAck_h tells the 21064 to sample data bytes 15..0 and the second non IDLE sample of dRAck_h tells the 21064 to sample data bytes 31..16. External logic may drive the second dRAck_h and the cAck_h during the same system clock.

READ_BLOCK and LDxL transactions may be terminated with HARD_ERROR status before all expected dRAck_h cycles are received. The contents of the entire cache block (including its tag and valid bit) are UNPREDICTABLE.

The 21064 may use D-stream primary cache fill data as soon as it is received, including data received in the first half of a READ_BLOCK transaction which is later terminated with HARD_ERROR. The 21064 does not use any I-stream primary cache fill data until it successfully receives the entire cache block.

The 21064 does not change its interpretation of dRAck_h 1..0 based on cAck_h if all expected dRAck's are received. External logic must avoid caching and/or ECC/parity checking data which is known to be garbage.

The 21064 behavior is UNDEFINED if dRAck_h is asserted in a non-read cycle.

The 21064 checks dRAck_h 0 (the bit that determines if the block is ECC/parity checked) during both halves of the 32-byte block. It is legal, but probably not useful, to check only one half of the block. The 21064 checks dRAck_h 1 during the first half of the 32-byte block.

The dOE_l inputs tells the 21064 if it should drive the data bus. It is a synchronous input, so external logic must guarantee setup and hold with respect to the system clock. If dOE_l is sampled true at a system clock, then the 21064 drives the data bus at the system clock if it has a WRITE_BLOCK or STxC request pending (the request may already be on the cReq pins, or it may appear on the cReq pins at the same system clock edge as the data appears). If dOE_l is sampled false at the system clock, then the 21064 tri-states the data bus on the next system clock cycle. The cycle type is factored into the enable so that systems can leave dOE_l asserted unless it is necessary to write a victim.

The dWSel_h inputs tells the 21064 which half of the 32-byte block of write data should be driven onto the data bus (dOE_l permitting). They are synchronous inputs. External logic must guarantee setup and hold with respect to the system clock. If dWSel_h 1 is sampled false at the end of a system clock cycle, then bytes 15..0 are driven onto the data bus in the next system clock cycle. If dWSel_h 1 is sampled true at the end of a system clock cycle, then

bytes 31..16 are driven onto the data bus in the next system clock cycle. Once dWSel_h 1 has been sampled true bytes 15..0 are lost. There is no backing up.

4.2.8 Primary Cache Invalidate

External logic needs to be able to invalidate primary data cache blocks to maintain coherence. The 21064 provides a mechanism to perform the necessary invalidates, but enforces no policy as to when invalidates are needed. Simple systems may choose to invalidate more or less blindly. Complex systems may choose to implement elaborate invalidate filters.

There are two situations where entries in the on-chip Dcache may need to be invalidated.

The first situation is the obvious one. Any time an external agent updates a block in memory (for example, an I/O device does a DMA transfer into memory) and the block has been loaded into the external cache, then the external cache block must be either invalidated or updated. If the external cache block has been loaded into the Dcache, then the Dcache block must be invalidated.

The second situation is more subtle. If a system is maintaining the Dcache as a subset of the external cache and an Icache miss results in an external cache block being replaced and that external cache block has been loaded into Dcache, then an invalidate is needed.

External logic invalidates an entry in the Dcache by asserting the dInvReq_h signal. The 21064 samples dInvReq_h at every system clock. When the 21064 detects dInvReq_h asserted, it invalidates the block in the Dcache whose index is on the iAdr_h pins.

The 21064 can accept an invalidate at every system clock.

The dInvReq_h input is synchronous. External logic must guarantee setup and hold with respect to the system clock. The iAdr_h inputs are also synchronous. External logic must guarantee setup and hold with respect to the system clock in any cycle in which dInvReq_h is true.

4.2.9 Interrupts

External interrupts are fed to the 21064 via the irq_h bus. The 6 interrupts are identical. They can be asynchronous, level sensitive, and individually masked by PALcode.

It is expected that on most systems, a combination of hardware and PALcode will use these 6 inputs as a power fail interrupt, a halt interrupt, and as 4 external interrupts (with the timer interrupt, the interprocessor interrupt, and the corrected read data interrupt wired to their normal IPL). This is not enforced by the 21064. Low-end systems could, for example, use all of them as device interrupts and arrange that its PALcode treated them all as IPL20 interrupts, using fixed vectors. See Section 2.3.3 for more details on interrupts.

To aid pattern-driven chip testers, the irq_h pins may be driven synchronously with respect to the system clock. See chapter Chapter 6 for the setup and hold requirements of the irq_h pins with respect to the system clock for this case.

4.2.10 Electrical Level Configuration

The 21064 can drive and receive either CMOS levels or 100K ECL levels (with assistance from resistors on the module).

The vRef input supplies a reference voltage to the input sense circuits. If external logic ties this to VSS + 1.4V, then all inputs sense TTL levels. If external logic ties this to VDD -1.3V (which can be obtained, for example, from the VBB output of an MC100E111) then all inputs sense ECL 100K levels.

The eclOut_h input selects the output levels. If external logic ties this false then all outputs generate CMOS levels. If external logic ties this true then all outputs are switched into a mode in which external resistors can be used to generate ECL 100K compatible levels.

4.2.11 Performance Monitoring

The perf_cnt_h 1..0 pins provide a means of giving the 21064's internal performance monitoring hardware access to off-chip events. These pins are system clock synchronous inputs which may be selected via the ICCSR IPR to be inputs to the performance counters inside the 21064 chip. If in a given system clock cycle a perf_cnt_h pin is sampled TRUE and the pin is selected as the source of its respective performance counter, then the counter will increment.

4.2.12 tristate

The tristate_l signal, if asserted, causes the 21064 to float all of its output and bidirectional pins with the exception of cpuClkOut_h. When tristate_l is asserted, The 21064 is forced into the reset state, but the irq_h pins are not resampled.

4.2.13 Continuity

The cont_l signal, if asserted, causes the 21064 to connect all of its pins to VSS, with the exception of clkIn_h, clkIn_l, testClkIn_h, testClkIn_l, cpuClkOut_h, sysClkOut1_h, sysClkOut1_l, sysClkOut2_h, sysClkOut2_l, VREF and cont_l.

4.3 64-Bit Mode

The 21064 may be configured at reset to use a 64-bit wide external data bus. In which case, data_ h 127..64 and check_h 27..14 are not used. These pins are internally pulled to VSS.

The dataA_h 3 pin is used as an additional address line for the external cache data RAMs. Like the dataA_h 4 pin, it should drive one input of a two input NOR gate, with the other input being driven by external logic. The 21064 drives dataA_h 3 false during reset, external cache hold, and any external cycle.

The dWSel_h 0 and dWSel_h 1 pins should be used by external logic to select which quadword of a 32-byte block is driven onto data_h 63..0 during each system clock cycle of an external WRITE_BLOCK or STxC transaction. The relationship between dWSel_h 1..0 and the selected bytes of the 32-block block is as follows:

Table 4–9: dWSel_h

dWSel_h 1..0	Selected Bytes
00	07..00
01	15..08
10	23..16
11	31..24

External logic must select quadwords in increasing order within the 32-byte block, but is free to skip over any quadword which does not have corresponding longword mask bits TRUE in cWMask_h 7..0.

Systems should ignore dataCEOE_h 3..2 and dataWE_h 3..2.

External cache read hit transactions are extended to consist of four cache read cycles in 64-bit mode. Each cache read cycle is (BC_RD_SPD + 1) CPU cycles in duration. The first cache read cycle consists of a tag probe and data read, while the subsequent three cache read cycles consist of data reads. The 21064 bus interface optimizes the external cache read hit transaction by wrapping cache read cycles around the quadword, which the 21064 originally requested. The dMapWE_h pin asserts 1 CPU cycle into the second cache read cycle and remains asserted until one CPU cycle before the end of the fourth cache read cycle.

External cache write hit transactions consist of one cache tag probe cycle, which is (BC_RD_SPD + 1) CPU cycles long and followed by one, two, three or four external cache write cycles which are each (BC_WR_SPD + 1) cycles long. The 21064 bus interface uses the minimum number of cache write cycles required to write the necessary longwords within the 32-byte block.

Note that the maximum latency from holdReq_h assertion to holdAck_h assertion in 64-bit mode is longer than in 128-bit mode. Also, the guarantee which external logic must make to the availability of the external cache data RAMs when asserting tagOk is different for 64-bit mode than for 128-bit mode.

For external READ_BLOCK and LDxL transactions the 21064 chip normally expects four distinct dRAck_h acknowledgment cycles. The first non-IDLE dRAck_h sample informs the 21064 to sample data bytes 7..0. The second to sample data bytes 15..8 and so on. Each quadword is parity/ECC checked based on the dRAck_h code supplied with that quadword. The dRAck_h code supplied with the first quadword determines whether the 32-byte block is cached.

4.4 Transactions

4.4.1 Reset

External logic resets the 21064 by asserting reset_l. When the 21064 detects the assertion of reset_l it terminates all external activity and places the output signals on the external interface into the following state. Note that all of the control signals have been placed in the state that allows external access to the external cache.

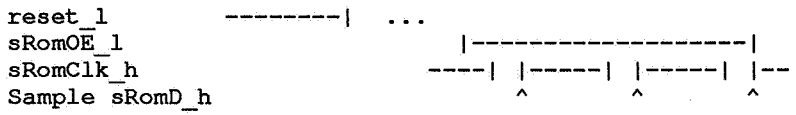
Table 4–10: Reset State

Pin	State
sRomOE_l	F
sRomClk_h	T
adr_h	Z
data_h	Z
check_h	Z
tagCEOE_h	F
tagCtlWE_h	F
tagCtlV_h	Z
tagCtlS_h	Z
tagCtlD_h	Z
tagCtlP_h	Z
dataCEOE_h	F
dataWE_h	F
dataA_h	F
holdAck_h	F
cReq_h 2:0	FFF

After asserting reset_l for long enough to reset the serial ROM (100 ns), external logic negates reset_l.

When the 21064 detects reset_l negate, it may load bits from an external serial ROM into its internal Icache, based on the value placed on icMode_h 1..0. The timing is shown below (assuming the 21064 only read 3 bits from the serial ROM):

Figure 4–4: 21064 RESET Sequence Timing



Each half-tick of the sRomClk_h signal is 63 CPU cycles long, which guarantees the 200ns clock high and clock low specifications and the 400ns clock to data specification of the serial ROM with 5ns CPU cycles.

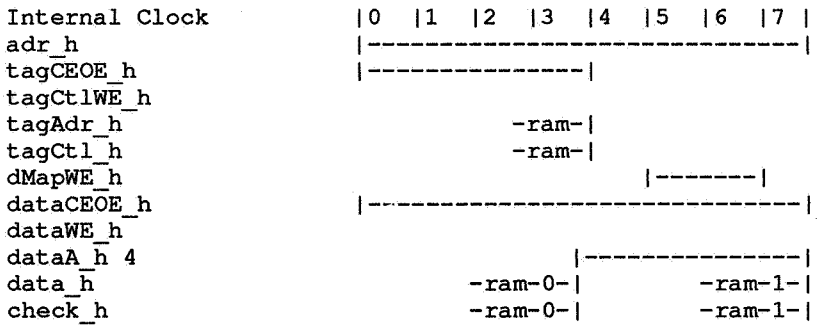
It is possible to disable the serial ROM mechanism altogether. See Section 4.2.3.

4.4.2 Fast External Cache Read Hit

A fast external cache read consists of a probe read (overlapped with the first data read), followed by the second data read if the probe hits.

The following diagram assumes that the external cache is using 4 cycle reads (BC_RD_SPD = 3), 4 cycle writes (BC_WR_SPD = 3), and chip enable control (OE = L).

Figure 4–5: Fast External Cache Read Sequence



If the probe misses then the cycle aborts at the end of clock 3.

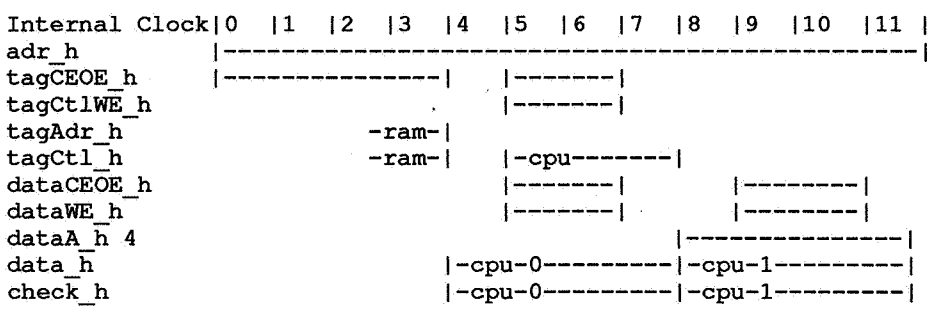
If the probe hits and the miss address had bit 4 set, then the two data reads would have been swapped (dataA_h 4 would have been true in cycles 0, 1, 2, 3, and would have been false in cycles 4, 5, 6, 7).

4.4.3 Fast External Cache Write Hit

A fast external cache write consists of a probe read, followed by 1 or 2 data writes.

The following diagram assumes that the external cache is using 4 cycle reads (BC_RD_SPD = 3), 4 cycle writes (BC_WR_SPD = 3), chip enable control (OE = L), and a 2 cycle write pulse centered in the 4 cycle write (BC_WE_CTL 15..1 = LLLLLLLLLLLLHH).

Figure 4–6: Fast External Cache Write Sequence



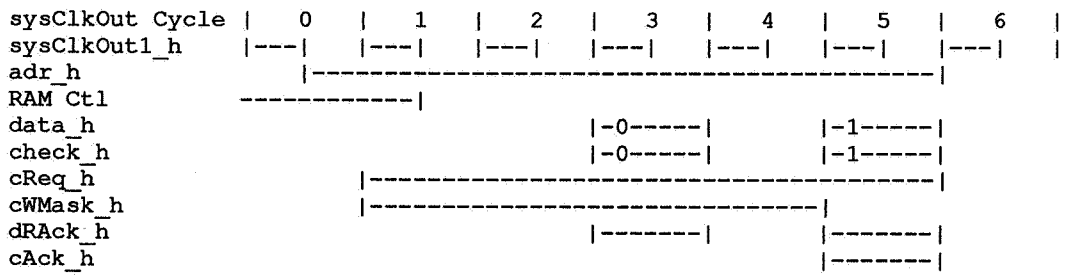
The 21064 drives the tagCtl_h pins one CPU cycle later than it drives the data_h and check_h pins, relative to the start of the write cycle. Unlike data_h and check_h, the tagCtl_h field must be read during the tag probe which proceeds the write cycle. Since the 21064 can switch its pins to a low impedance state much more quickly than most RAMs can switch their pins to a high impedance state, the 21064 waits one CPU cycle before driving the tagCtl_h pins in order to minimize three state driver overlap.

If the probe misses then the cycle aborts at the end of clock 3.

4.4.4 READ_BLOCK Transaction

A READ_BLOCK transaction appears at the external interface on external cache read misses, either because it really was a miss or because the external cache has not been enabled.

Figure 4–7: READ_BLOCK Sequence



0. The cReq_h pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The adr_h pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.
1. The READ_BLOCK transaction begins. The 21064 has already placed the address of the block containing the miss on adr_h. The 21064 places the quadword-within-block and the I/D indication on cWMask_h. The 21064 places a READ_BLOCK command code on cReq_h. The 21064 will clear the RAM control pins (dataA_h 4..3, dataCEOE_h 3..0 and tagCEOE_h) no later than one CPU cycle after the system clock edge at which the transaction begins.
2. The external logic obtains the first 16 bytes of data. Although a single stall cycle has been shown, there could be no stall cycles or many stall cycles.
3. The external logic has the first 16 bytes of data. It places it on the data_h and check_h busses. It asserts dRAck_h to tell the 21064 that the data and check bit busses are valid. The 21064 detects dRAck_h at the end of this cycle and reads in the first 16 bytes of data at the same time.
4. The external logic obtains the second 16 bytes of data. Although a single stall cycle has been shown, there could be no stall cycles or many stall cycles.

5. The external logic has the second 16 bytes of data. It places it on the data_h and check_h busses. It asserts dRAck_h to tell the 21064 that the data and check bit busses are valid. The 21064 detects dRAck_h at the end of this cycle and reads in the second 16 bytes of data at the same time. The external logic places an acknowledge code on cAck_h to tell the 21064 that the READ_BLOCK cycle is completed. The 21064 detects the acknowledge at the end of this cycle and can change the address.

6. Everything is idle. The 21064 could start a new external cache cycle at this time.

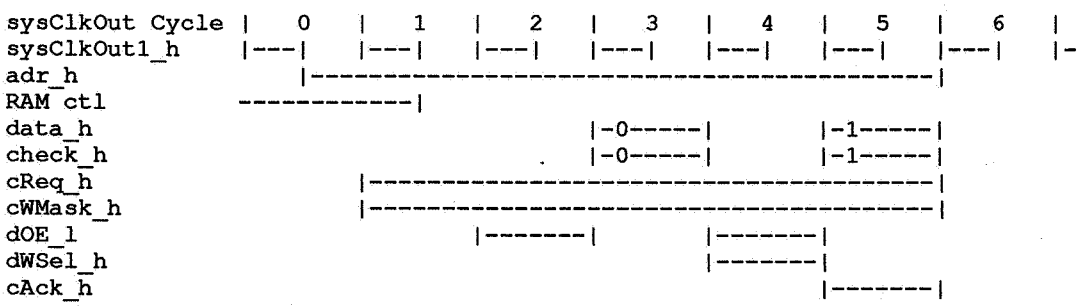
External logic owns the RAMs by virtue of 21064 having deasserted its RAM control signals at the beginning of the transaction, external logic may cache the data by asserting its write pulses on the external cache during cycles 3 and 5.

The 21064 performs ECC checking and correction (or parity checking) on the data supplied to it via the data and check busses, if requested by the acknowledge code. It is not necessary to place data into the external cache to get checking and correction.

4.4.5 Write Block

A WRITE_BLOCK transaction appears at the external interface on external cache write misses (either because it really was a miss or because the external cache has not been enabled) or external cache write hits to shared blocks.

Figure 4–8: WRITE_BLOCK Sequence



0. The cReq_h pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The adr_h pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.

1. The WRITE_BLOCK cycle begins. The 21064 has already placed the address of the block on adr_h. The 21064 places the longword valid masks on cWMask_h and a WRITE_BLOCK command code on cReq_h. The 21064 will clear dataA_h 4..3 and tagCEOE_h no later than one CPU cycle after the system clock edge, at which time the transaction begins. The 21064 clears dataCEOE_H 3..0 at least one CPU cycle before the system clock edge, at which time the transaction begins.

2. The external logic detects the command and asserts dOE_l to tell the 21064 to drive the first 16 bytes of the block onto the data bus. The timing shown for dOE_l is chosen for discussion purposes. External logic can assert dOE_l by default and only deassert it when it needs to read the data RAMs, such as when writing back a victim block.
3. The 21064 drives the first 16 bytes of write data onto the data_h and check_h busses and the external logic writes it into the destination. Although a single stall cycle has been shown, there could be no stall cycles or many stall cycles.
4. The external logic asserts dOE_l and dWSel_h to tell the 21064 to drive the second 16 bytes of data onto the data bus.
5. The 21064 drives the second 16 bytes of write data onto the data_h and check_h busses and the external logic writes it into the destination. Although a single stall cycle has been shown, there could be no stall cycles or many stall cycles. External logic places an acknowledge code on cAck_h to tell the 21064 that the WRITE_BLOCK cycle is completed. The 21064 detects the acknowledge at the end of this cycle and changes the address and command to their next values.
6. Everything is idle. The 21064 can start a new external cache access at this time.

External logic owns the RAMs by virtue of the 21064 having deasserted its RAM control signals at the beginning of the transaction, external logic may cache the data by asserting its write pulses on the external cache during cycles 3 and 5.

The 21064 performs ECC generation (or parity generation) on data it drives onto the data bus.

Although in the above diagram external logic cycles through both 128-bit chunks of potential write data, this need not always be the case. External logic must pull from the 21064 chip only those 128-bit chunks of data which contain valid longwords as specified by the cWMask_h signals. The only requirement is that if both halves are pulled from the 21064, then the lower half must be pulled before the upper half.

4.4.6 LDxL Transaction

An LDxL transaction appears at the external interface when an interlocked load instruction is executed. The external cache is not probed. With the exception of the command code output on the cReq pins, the LDxL transaction is exactly the same as a READ_BLOCK transaction. See section Section 4.4.4.

4.4.7 STxC Transaction

An STxC transaction appears at the external interface when a conditional store instruction is executed. The external cache is not probed.

The STxC transaction is the same as the WRITE_BLOCK transaction, with the following exceptions:

0. The code placed on the cReq pins is different.
1. The cWMask field will never validate more than a single longword or quadword of data.

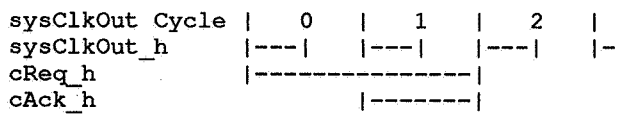
2. External logic has the option of making the transaction fail by using the cAck code of STxC_FAIL. It may do so without asserting either dOE_l or dWSel_h.

See section Section 4.4.5.

4.4.8 BARRIER Transaction

A BARRIER transaction appears on the external interface as a result of an MB instruction. The acknowledgment of the BARRIER transaction tells the 21064 that all invalidates have been supplied to it, and that any external write buffers have been pushed out to the coherence point. Any errors detected during these operations can be reported to the 21064 when the BARRIER transaction is acknowledged.

Figure 4–9: BARRIER Request/Acknowledge Sequence

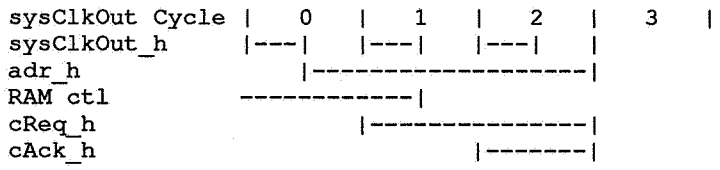


0. The BARRIER transaction begins. The 21064 places the command code for BARRIER onto the cReq_h outputs.
1. The external logic notices the BARRIER command and places an acknowledge code on the cAck_h inputs.
2. The 21064 detects the acknowledge on cAck_h and removes the command. The external logic removes the acknowledge code from cAck_h. The cycle is finished.

4.4.9 FETCH Transaction

A FETCH transaction appears on the external interface as a result of a FETCH instruction. The transaction supplies an address to the external logic, which can choose to ignore it or use it as a memory-to-cache prefetching hint.

Figure 4–10: FETCH Sequence



0. The cReq_h pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The adr_h pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.
1. The FETCH transaction begins. The 21064 has already placed the effective address of the FETCH on the address outputs. The 21064 places the command code for FETCH on the cReq_ h outputs. The 21064 will clear the RAM control pins (dataA_h 4..3, dataCEOE_h 3..0 and tagCEOE_h) no later than one CPU cycle after the system clock edge, at which time the transaction begins.
2. The external logic notices the FETCH command and places an acknowledge code on the cAck_h inputs.
3. The 21064 detects the acknowledge on cAck_h and removes the address and the command. The external logic removes the acknowledge code from cAck_h. The cycle is finished.

4.4.10 FETCHM Transaction

A FETCHM transaction appears on the external interface as a result of a FETCHM instruction. The transaction supplies an address to the external logic, which can choose to ignore it or use it as a memory-to-cache prefetching hint. With the exception of the command code placed on cReq_h, the FETCHM transaction is the same as the FETCH transaction. See section Section 4.4.9.

Chapter 5
DC Characteristics

5.1 Overview

The 21064 is capable of running in a CMOS/TTL environment or an ECL environment. The chip will be tested and characterized in a CMOS environment. The specifications below assume a CMOS/TTL environment. Differences for an ECL environment are noted in Section 5.2.

5.1.1 Power Supply

In CMOS mode the VSS pins are connected to 0.0V, and the VDD pins are connected to 3.3V, +/- 5%.

To prevent damage to the chip, it is important that the 3.3V power supply be stable before any input or bidirectional pins be allowed to rise above 4.0V.

To help in meeting this requirement, the assertion levels of the 21064's input pins have been arranged so that their default state is the electrical low state. This makes them active high, with the exception of tagOk_l and dOE_l, which are true (low) by default.

5.1.2 Reference Supply

The vRef analog input should be connected to a 1.4V +/-10% reference supply.

5.1.3 Input Clocks

The clkIn_h,_l signals are differential signals generated from an ECL oscillator circuit, although non-ECL circuits can also be used. The signals can be AC coupled, with a nominal DC bias of VDD/2 set by a high-impedence (i.e. >1K) resistive network on the chip. The signals need not be AC coupled if VDD is used as the VCC supply to the ECL oscillator. See the AC Characteristics chapter for more detail.

5.1.4 Signal pins

Input pins are ordinary CMOS inputs with standard TTL levels, see Table 5–1. Once power has been applied and vRef has met its hold time, the majority of input pins can be driven by 5.0V (nominal) signals without harming the 21064. There are some signals that are sampled before vRef is stable, and these signals can not be driven above the power supply. These signals are:

- dcOk_h
- tristate_l
- cont_l
- eclOut_h

Output pins are ordinary 3.3V CMOS outputs. Although output signals are rail-to-rail, timing is specified to standard TTL levels, see Table 5–1.

Bidirectional pins are ordinary 3.3V CMOS bidirectional. On input, they act like input pins. On output, they drive like output pins.

Once power has been applied, input (except noted above) and bidirectional pins can be driven to a maximum DC voltage of 5.5V without harming the 21064 (it is not necessary to use static RAMS with 3.3V outputs).

Table 5–1: CMOS DC Characteristics - TTL Inputs/Outputs

Symbol	Description	Min	Max	Units	Test Cond.
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 = -100uA
Vol	Low level output voltage		0.4	V	Iol = 3.2mA
Icin	Clock input Leakage	4	4	mA	-0.5<Vin<3.6V
Iil	Input leakage current	10	10	uA	0<Vin<Vdd V
Iol	Output leakage current	-10	-10	uA	

5.2 ECL 100K Mode

In ECL 100K mode a combination of on-chip and off-chip circuits provide ECL 100K compatible interfaces.

5.2.1 Power Supply

In ECL 100K mode the VDD pins are connected to 0.0V, and the VSS pins are connected to -3.3V, +/- 5%.

5.2.2 Reference Supply

In ECL 100K mode the vRef input is connected to a reference supply at VDD minus 1.3V. The best way to generate the reference supply is to use the VBB output provided by several chips, such as the ECLinPS MC100E111.

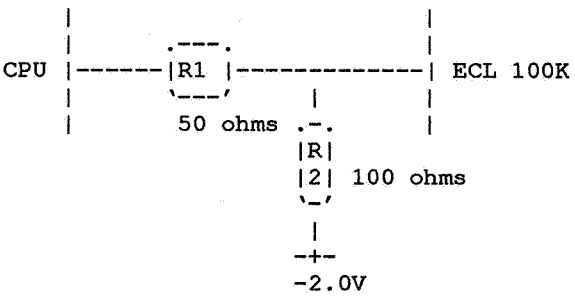
5.2.3 Inputs

In ECL 100K mode inputs appear to be ordinary ECL 100K inputs, with the exception that they lack the pull down resistor that is normally present in ECL 100K circuits.

5.2.4 Outputs

In ECL 100K mode external resistors create the correct ECL 100K levels. The following stylized circuit is used.

Figure 5–1: ECL Termination



5.2.5 Bidirectionals

In ECL 100K mode the bidirectional pins should be converted into unidirectional input and output busses as close to the 21064 as possible. The 21064 chip bidirectional bus is buffered and driven onto the system output bus. The system input bus is driven onto the 21064's bidirectional bus using cut-off drivers controlled by the CPU's output enables.

The same resistor network used on output pins is used on bidirectional pins.

5.3 Power Dissipation

A comprehensive power dissipation analysis was performed using a program that caused maximum dynamic power dissipation. It was run on a logic simulator and using analysis tools, power dissipation was analytically predicted. The results from that analysis are shown in Table 5–2.

Table 5–2: 21064 Power Dissipation @Vdd=3.45V

Speed	Min	Typ	Max	Units
5.0ns	24	29.5	36	Watts
6.6ns	19	23	27.5	Watts

The minimum power occurs during reset, the "Typ" column is the worst case average program and the "Max" column is the worst case pathological program. An important observation is the fact that all normal programs (both stand-alone and under a high level operating system) run in a range between "Min" and "Typ". So while the pathological case is theoretically possible, it is extremely unlikely in practice. The following approach is recommended for system designers:

- Design the heat sink and thermal environment to keep the die temperature to 85C for the "Typ" power case. This is certainly the limiting case for average power dissipation to be used for long term reliability assessment. With "Typ" designed for 85C, then in all cases, "Max" will result in die temperatures under the 100C design, test and process qual limits.
- Design the overall enclosure and the power supply to handle the "Max" power case.

It is possible to account for applications where the maximum supply voltage is other than 3.45V and/or the operating frequency is not 150 or 200MHz. The formulae for calculating Idd under various conditions are as follows:

Idd (Min) = 116mA/V + 9.6mA/V*MHz
Idd (Typ) = 116mA/V + 11.7mA/V*MHz
Idd (Max) = 116mA/V + 14.4mA/V*MHz

Chapter 6
AC Characteristics

This chapter contains the AC specification for the 21064-AA. Timing parameters are given for an internal clock speed of 150 Mhz (6.6ns cycle time).

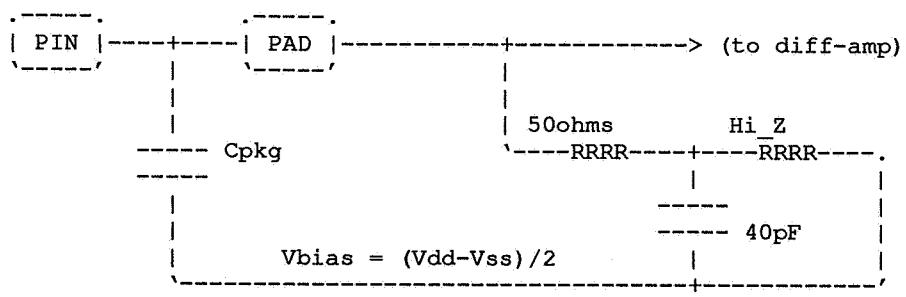
6.1 vRef

vRef is an analog reference voltage used by the input buffers of all signals except clkIn_h,_l, testclkIn_h,_l, tagOk_h,_l, dcOk_h, eclOut_h, tristate_l, and cont_l. Upon power up, reset_l cannot be sampled until vRef is stable. There is a large internal capacitance on vRef and an RC delay between its pin and the input buffers. Therefore, systems must not assert dcOk_h until a suitable interval following the stability of the vRef source. This interval is specified as the greater of 1us and 10nF * Zout, where Zout is the vRef source impedance.

6.2 Input Clocks

The input clocks clkIn_h,_l and testclkIn_h,_l are received differentially, then XORed to provide the time-base when dcOk_h is asserted. The testclkIn_h, _lsignals should only be used by testers unable to drive clkIn_h, _l at full speed. The terminations on these signals are designed to be compatible with system oscillators of arbitrary DC bias. Schematically, they look as follows:

Figure 6–1: Clock Termination



This is designed to approximate a 50ohm termination for the purpose of impedance matching for those systems (if any) which drive input clocks across long traces. Furthermore, the high impedance bias driver allows a clock source of arbitrary DC bias to be AC coupled to the clock input. The peak-to-peak amplitude of the clock source must be between 0.6V and 3.0V as seen by the 21064. Either a "square-wave" or a sinusoidal source can be used.

Note that full-rail clocks can be driven by testers.

The following table lists the input clock cycle times. Note that these periods equal one-half the corresponding cpu cycle times.

Table 6–1: Input Clock Timing

Name	21064-AA	Unit
clkIn period min	3.3	ns
clkIn period max	tbd	ns
clkIn symmetry	50%+/-10%	percent

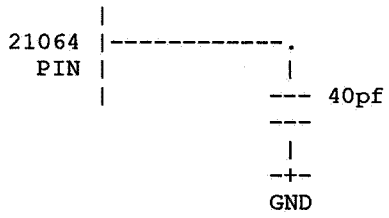
6.3 cpuClkOut_h

The cpuClkOut_h signal is expected to be used only by an ECL synchronizer in systems using the tagOk protocol. In order to accommodate ECL levels, the driver consists of only a PMOS pullup device. ECL 100K levels may be constructed with a 50ohm board resistor in series with the driver and a 100ohm board resistor between the load and Vdd minus 2V. CMOS Vdd must equal ECL Vcc in this scheme. Note that the trace must be short to insure good signal integrity if the board impedance is not in the vicinity of 100ohm.

6.4 Test Configuration

All outputs and bidirectional signals including clocks (but excluding cpuClkOut_h) are specified with respect to a standard 40pF load as shown below. All timing is specified with respect to the crossing of standard TTL input levels at 0.8V and 2.0V.

Figure 6–2: Standard Load



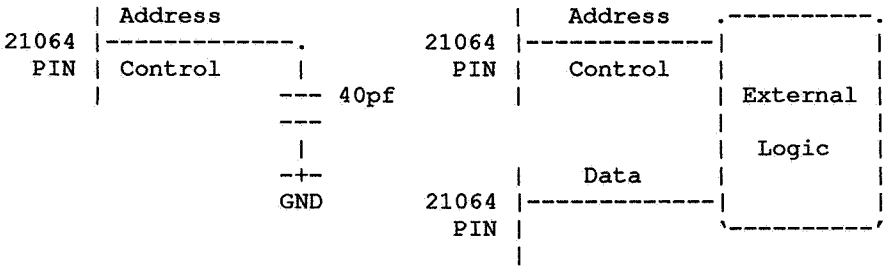
6.5 Fast Cycles on External Cache

From a system standpoint, fast cycles on the external cache are completely unclocked. The two cases of read and write cycles require separate treatment.

6.5.1 Fast Read Cycles

External logic must meet the maximum flow-through delay, as defined with respect to the circuits below.

Figure 6–3: Flow-through Delay



"Address" refers to adr_h and dataA_h. "Control" refers to dataCEOE_h and tagCEOE_h. "Data" refers to data_h, check_h, tagAdr_h, and tagCtl_h. Assume that address/control is driven from the same internal clock edge in the two cases above. External flow-through delay is defined as the delay between address/control valid to the 40pF standard load in the case on the left and data valid to the 21064 in the case on the right. It can not exceed the fast read cycle time (i.e. BC_RD_SPD+1 cpu cycles) less 5.0ns. The 21064 guarantees that its address drivers are enabled at least one cpu cycle prior to a fast cache access, such that, adr_h never needs to be pulled down from 5V during the cycle.

6.5.2 Fast Write Cycles

External logic must guarantee that fast write cycles complete. Data, address, and control (including dataWE_h and tagCtlWE_h) are driven by the 21064 with identical timing from its internal clock. Actual pulse widths are at least the nominal width less 1.5ns or 2.9ns on lines precharged to 5V (i.e. data lines following a probe read). The timing of dMapWE_h during dcache read hits is specified in the same way.

6.6 External Cycles

All external cycle timing is referenced to the rising edge of sysClkOut1_h. Input setup, hold times, output delay, and enable times are referenced to the point at which sysClkOut1_h crosses 0.8V. (Output enable time is defined as output delay time from a tri- stated state. It can differ from the nominal delay, because it may entail pulling the signal down from a 5V level.) Output hold times are referenced to the point at which sysClkOut1_h crosses 2.0V. They denote the times beyond sysClkOut1_h for which outputs hold their valid values from the previous cycle. Note that these times are negative, meaning that data may lose validity BEFORE sysClkOut1_h becomes valid high. (This is possible because there is no cause-effect relationship between the system clock outputs and data. The system clock outputs are nothing more than data pins which happen to switch in a fixed pattern.) Address enable timing is relevant only for systems using the holdReq protocol with two cpu cycles per system cycle. All bidirectional lines can be considered enable or disabled simultaneously with the rising edge of sysClkOut1_h.

Table 6–2: External Cycles

Name	Min	Max	Units
Enable, sysClkOut1_h to			
adr_h, data_h, check_h		2.9	ns
Output Delay, sysClkOut1_h to			
adr_h, data_h, check_h, cReq_h, cWMask_h, holdAck_h		+1.5	ns
Output hold, sysClkOut1_h to			
adr_h, data_h, check_h, cReq_h, cWMask_h, holdAck_h	-1.5		ns
Input Setup relative to sysClkOut1_h			
dRAck_h, dWSel_h, dOE_l	9.3		ns
cAck_h	Tcyc/2+6.0		ns
holdReq_h	4.8		ns
dInvReq_h, iAdr_h, , perf_cnt_h	4.5		ns
data_h, check_h	3.5		ns
Input Hold relative to sysClkOut1_h			
cAck_h, dRAck_h, dWSel_h, dOE_l	0		ns
data_h, check_h	0		ns
holdReq_h, dInvReq_h, iAdr_h	0		ns

The cAck_h input setup time is a function of the chip cycle time(Tcyc). At the nominal 6.6ns cycle time, required setup on the cAck_h pin is 9.3ns.

6.7 tagEq

When active during external cache hold, the timing of tagEq_l is specified from the time its inputs become valid at the pins.

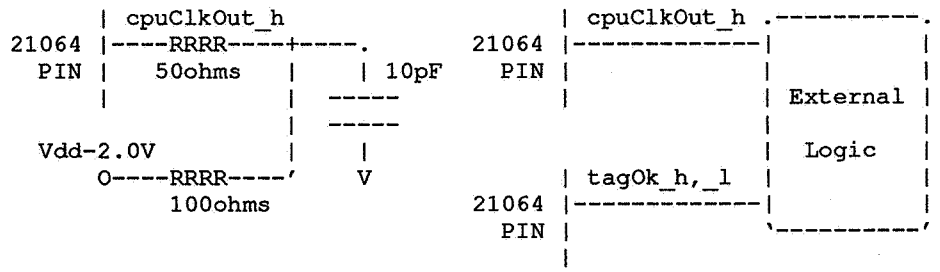
Table 6–3: tagEq

Name	Min	Max	Units
Delay, adr_h -> tagEq_l		17.0	ns
Delay, tagAdr_h -> tagEq_l		17.0	ns

6.8 tagOk

The tagOk_h,_l signals are expected to be driven to the 21064 directly from the final stage of an ECL synchronizer clocked by cpuClkOut_h. As in the case of fast external cache cycles, the system must meet a maximum flow-through delay. This delay is defined with respect to the circuits below.

Figure 6–4: Flow-Through Delay



Assume that cpuClkOut_h is driven from the same internal clock edge in the two cases above. External flow-through delay is defined as the delay between cpuClkOut_h valid to the 10pF ECL "standard" load in the case on the left and tagOK_h,_l valid to the 21064 in the case on the right. It can not exceed the nominal cpu cycle time less 3.9ns. Note that board resistors must be part of "external logic" in the circuit on the right. For purposes of this specification, cpuClkOut_h is considered valid when it crosses the ECL threshold "Vbb" (equal to roughly Vcc - 1.3V) and tagOk is considered valid when the differential lines cross each other.

6.9 Tester Considerations

6.9.1 Inputs

The signals reset_l, irq_h, and sRomD_h (in serial port mode) are asynchronous during normal system operation. However, for test purposes they should be driven synchronously with sysClkOut1_h with the timing given below. Note that these parameters are given with respect to the time at which the rising edge of sysClkOut1_h crosses 0.8V.

Table 6–4: Asynchronous Signals on a Tester

Name	Min	Max	Units
Setup, reset_l -> sysClkOut1_h	5.0		ns
Setup, irq_h -> sysClkOut1_h	5.0		ns
Hold, irq_h -> sysClkOut1_h	0		ns
Setup, sRomD_h -> sysClkOut1_h	5.0		ns
Hold, sRomD_h -> sysClkOut1_h	0		ns

6.9.2 Signals Timed from Cpu Clock

Due to the speed of the 21064, it is expected that at-speed testing will be done with tester cycle equal to system cycle (i.e. sysClkOut1_h). However, fast external cache operation and serial ROM operation are timed from internal cpu clock. Therefore, input sampling, output enabling, and switching can occur at different time points within a tester cycle from one cycle to the next. Fortunately, the number of such points is finite, equal to the number of cpu cycles per tester cycle. For any given transaction, each signal will have its standard external cycle timing with respect to the rising edge of sysClkOut1_h OR to a "phantom" edge offset from sysClkOut1_h by exactly an integer number of cpu cycles. (Note that dataA_h, dataCEOE_h, dataWE_h, tagCEOE_h, tagCtlWE_h, and dMapWE_h have the same delay timing as adr_ h.) Therefore, outputs may be sampled deterministically with appropriate placement of the tester strobe and inputs may be received deterministically with appropriate placement of the drive edge. Bidirectional signals present a different problem. Because the tester can enable or disable a given driver at just one point within its cycle, it must in the worst case drive an input beyond its 21064 sample point by at least (N-1) cpu cycles, where N is the number of cpu cycles per system cycle. However, in the worst case the 21064 will enable its drivers just one cpu cycle after sampling (for example, tagCtl_h following probe write). Therefore, the number of cpu cycles per system cycle must not exceed two to avoid driver conflict between the 21064 and the tester.

The serial ROM outputs sRomOE_l and sRomClk_h can be strobed with the same timing as the data_h pins when driven by the 21064. The serial ROM input sRomD_h can be switched with the same timing used in serial port mode.

Chapter 7
Package Information

Figure 7–1 shows the package physical dimensions without heat sink. Figure 7–2 shows pin locations.

Figure 7–1: Package Dimensions

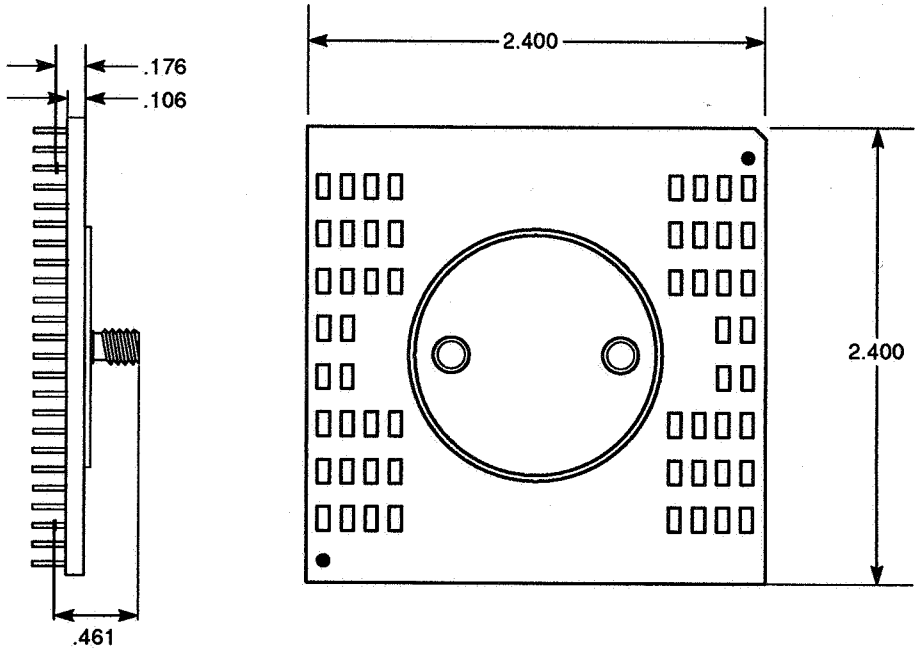
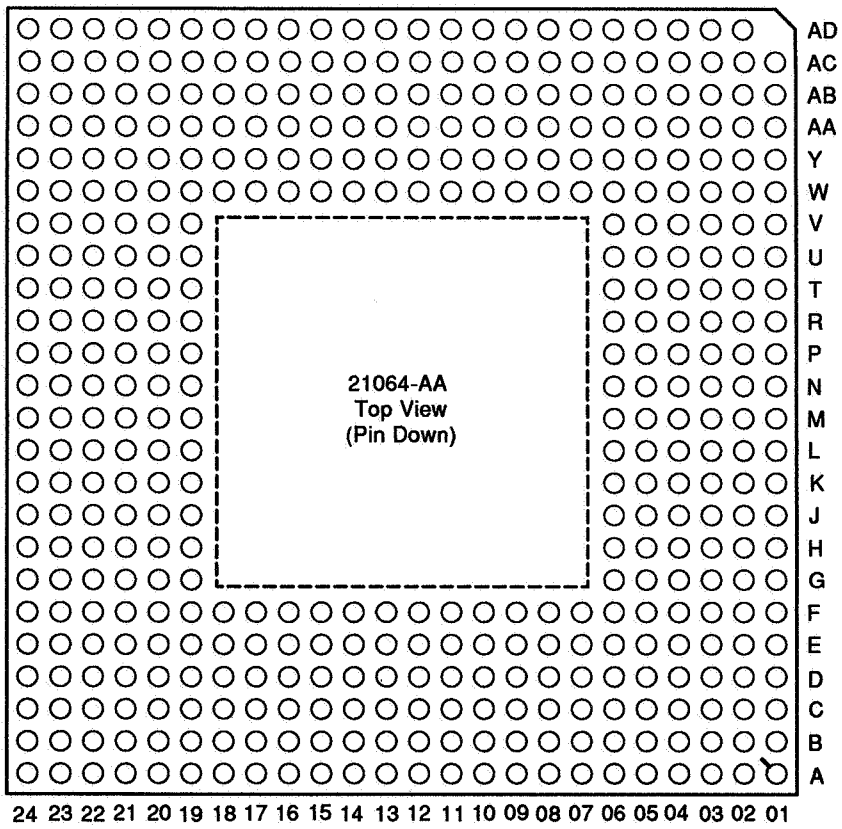


Figure 7–2: PGA Cavity Down View



Chapter 8
Pinout

8.1 Overview

This chapter contains the entire 21064 pinout. The order is by PGA location. Int the "TYPE" column, B = Bidirectional, I = Input, N = Not connected, P = Power or Ground, and O = Output.

8.2 21064 Pinout

Table 8–1: 21064 Pin List

PGA Loc.	PIN No.	Type	Name
A1	001	B	data_h 33
A2	002	B	data_h 97
A3	003	B	data_h 98
A4	004	B	data_h 100
A5	005	B	data_h 38
A6	006	B	check_h 27
A7	007	B	data_h 104
A8	008	B	data_h 42
A9	009	B	data_h 44
A10	010	B	data_h 109
A11	011	B	data_h 47
A12	012	B	data_h 49
A13	013	B	data_h 113

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
A14	014	B	data_h 52
A15	015	B	check_h 12
A16	016	B	data_h 55
A17	017	B	data_h 120
A18	018	B	data_h 122
A19	019	B	check_h 7
A20	020	B	data_h 60
A21	021	B	data_h 61
A22	022	B	data_h 62
A23	023	B	data_h 127
A24	024	B	check_h 9
B1	025	B	check_h 15
B2	026	P	VDD plane
B3	027	B	data_h 35
B4	028	P	VSS plane
B5	029	B	data_h 101
B6	030	P	VDD plane
B7	031	B	data_h 40
B8	032	P	VSS plane
B9	033	B	data_h 107
B10	034	P	VDD plane
B11	035	B	data_h 110
B12	036	P	VSS plane
B13	037	B	data_h 50
B14	038	P	VDD plane
B15	039	B	check_h 26
B16	040	P	VSS plane
B17	041	B	data_h 57
B18	042	P	VDD plane

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
B19	043	B	check_h 21
B20	044	P	VSS plane
B21	045	B	data_h 125
B22	046	P	VDD plane
B23	047	P	VSS plane
B24	048	B	check_h 8
C1	049	B	check_h 16
C2	050	P	VSS plane
C3	051	B	data_h 96
C4	052	B	data_h 99
C5	053	B	data_h 37
C6	054	B	check_h 13
C7	055	B	data_h 103
C8	056	B	data_h 105
C9	057	B	data_h 43
C10	058	B	data_h 45
C11	059	B	data_h 46
C12	060	B	data_h 112
C13	061	B	data_h 114
C14	062	B	data_h 116
C15	063	B	data_h 54
C16	064	B	data_h 119
C17	065	B	data_h 121
C18	066	B	check_h 11
C19	067	B	data_h 59
C20	068	B	data_h 124
C21	069	B	data_h 126
C22	070	B	check_h 23
C23	071	I	dRAck_h 0

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
C24	072	N	spare 3
D1	073	B	data_h 94
D2	074	B	check_h 2
D3	075	B	check_h 1
D4	076	B	data_h 34
D5	077	B	data_h 36
D6	078	B	data_h 102
D7	079	B	data_h 39
D8	080	B	data_h 41
D9	081	B	data_h 106
D10	082	B	data_h 108
D11	083	B	check_h 24
D12	084	B	data_h 48
D13	085	B	data_h 51
D14	086	B	data_h 53
D15	087	B	data_h 118
D16	088	B	data_h 56
D17	089	B	data_h 58
D18	090	B	check_h 25
D19	091	B	data_h 123
D20	092	B	data_h 63
D21	093	B	check_h 22
D22	094	I	dRAck_h 2
D23	095	P	VDD plane
D24	096	I	dOE_l
E1	097	B	data_h 30
E2	098	P	VDD plane
E3	099	B	data_h 31
E4	100	B	data_h 32

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
E5	101	P	VDD plane
E6	102	P	VSS plane
E7	103	P	VDD plane
E8	104	P	VSS plane
E9	105	P	VDD plane
E10	106	P	VSS plane
E11	107	B	check_h 10
E12	108	B	data_h 111
E13	109	B	data_h 115
E14	110	B	data_h 117
E15	111	P	VDD plane
E16	112	P	VSS plane
E17	113	P	VDD plane
E18	114	P	VSS plane
E19	115	P	VDD plane
E20	116	P	VSS plane
E21	117	I	dRAck_h 1
E22	118	I	dWSel_h 0
E23	119	I	dWSel_h 1
E24	120	I	cAck_h 0
F1	121	B	data_h 92
F2	122	B	data_h 29
F3	123	B	data_h 93
F4	124	B	data_h 95
F5	125	P	VSS plane
F6	126	P	VDD plane
F7	127	P	VSS plane
F8	128	P	VDD plane
F9	129	P	VSS plane

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
F10	130	P	VDD plane
F11	131	P	VSS plane
F12	132	P	VDD plane
F13	133	P	VSS plane
F14	134	P	VDD plane
F15	135	P	VSS plane
F16	136	P	VDD plane
F17	137	P	VSS plane
F18	138	P	VDD plane
F19	139	P	VSS plane
F20	140	P	VDD plane
F21	141	I	cAck_h 1
F22	142	I	cAck_h 2
F23	143	P	VSS plane
F24	144	I	holdReq_h
G1	145	B	data_h 27
G2	146	P	VSS plane
G3	147	B	data_h 91
G4	148	B	data_h 28
G5	149	P	VDD plane
G6	150	P	VSS plane
G19	151	P	VDD plane
G20	152	P	VSS plane
G21	153	O	holdAck_h
G22	154	O	dataCEOE_h 0
G23	155	O	dataCEOE_h 1
G24	156	O	dataCEOE_h 2
H1	157	B	check_h 4
H2	158	B	check_h 18

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
H3	159	B	check_h 0
H4	160	B	check_h 14
H5	161	P	VSS plane
H6	162	P	VDD plane
H19	163	P	VSS plane
H20	164	P	VDD plane
H21	165	O	dataCEOE_h 3
H22	166	O	tagCtlWE_h
H23	167	P	VDD plane
H24	168	O	cWMask_h 0
J1	169	B	data_h 89
J2	170	P	VDD plane
J3	171	B	data_h 26
J4	172	B	data_h 90
J5	173	P	VDD plane
J6	174	P	VSS plane
J19	175	P	VDD plane
J20	176	P	VSS plane
J21	177	O	cWMask_h 1
J22	178	O	cWMask_h 2
J23	179	O	cWMask_h 3
J24	180	O	cWMask_h 4
K1	181	B	data_h 87
K2	182	B	data_h 24
K3	183	B	data_h 88
K4	184	B	data_h 25
K5	185	P	VSS plane
K6	186	P	VDD plane
K19	187	P	VSS plane

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
K20	188	P	VDD plane
K21	189	O	cWMask_h 5
K22	190	O	cWMask_h 6
K23	191	P	VSS plane
K24	192	O	cWMask_h 7
L1	193	B	check_h 19
L2	194	P	VSS plane
L3	195	B	data_h 22
L4	196	B	data_h 86
L5	197	B	data_h 23
L6	198	P	VSS plane
L19	199	P	VDD plane
L20	200	O	dataWE_h 0
L21	201	O	dataWE_h 1
L22	202	O	dataWE_h 2
L23	203	O	dataWE_h 3
L24	204	O	dMapWE_h
M1	205	B	data_h 20
M2	206	B	data_h 84
M3	207	B	data_h 21
M4	208	B	data_h 85
M5	209	B	check_h 5
M6	210	P	VDD plane
M19	211	P	VSS plane
M20	212	O	cReq_h 0
M21	213	O	cReq_h 1
M22	214	O	cReq_h 2
M23	215	P	VDD plane
M24	216	N	spare 0

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
N1	217	B	data_h 83
N2	218	P	VDD plane
N3	219	B	data_h 19
N4	220	B	data_h 82
N5	221	B	data_h 18
N6	222	P	VSS plane
N19	223	P	VDD plane
N20	224	I	tagOk_l
N21	225	I	tagOk_h
N22	226	O	dataA_h 4
N23	227	O	dataA_h 3
N24	228	O	tagCEOE_h
P1	229	B	data_h 81
P2	230	B	data_h 17
P3	231	B	data_h 80
P4	232	B	data_h 16
P5	233	B	data_h 79
P6	234	P	VDD plane
P19	235	P	VSS plane
P20	236	B	tagCtlS_h
P21	237	B	tagCtlD_h
P22	238	B	tagCtlP_h
P23	239	P	VSS plane
P24	240	O	tagEq_l
R1	241	B	data_h 15
R2	242	P	VSS plane
R3	243	B	data_h 78
R4	244	B	data_h 14
R5	245	P	VDD plane

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
R6	246	P	VSS plane
R19	247	P	VDD plane
R20	248	P	VSS plane
R21	249	I	tagadr_h 19
R22	250	I	tagadr_h 18
R23	251	I	tagadr_h 17
R24	252	B	tagCtlV_h
T1	253	B	check_h 17
T2	254	B	check_h 3
T3	255	B	data_h 77
T4	256	B	data_h 13
T5	257	P	VSS plane
T6	258	P	VDD plane
T19	259	P	VSS plane
T20	260	P	VDD plane
T21	261	I	tagadr_h 22
T22	262	I	tagadr_h 21
T23	263	P	VDD plane
T24	264	I	tagadr_h 20
U1	265	B	data_h 76
U2	266	P	VDD plane
U3	267	B	data_h 12
U4	268	B	data_h 75
U5	269	P	VDD plane
U6	270	P	VSS plane
U19	271	P	VDD plane
U20	272	P	VSS plane
U21	273	I	tagadr_h 26
U22	274	I	tagadr_h 25

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
U23	275	I	tagadr_h 24
U24	276	I	tagadr_h 23
V1	277	B	data_h 11
V2	278	B	data_h 74
V3	279	B	data_h 10
V4	280	B	data_h 73
V5	281	P	VSS plane
V6	282	P	VDD plane
V19	283	P	VSS plane
V20	284	P	VDD plane
V21	285	I	tagadr_h 29
V22	286	I	tagadr_h 28
V23	287	P	VSS plane
V24	288	I	tagadr_h 27
W1	289	B	data_h 9
W2	290	P	VSS plane
W3	291	B	data_h 72
W4	292	B	check_h 6
W5	293	P	VDD plane
W6	294	P	VSS plane
W7	295	P	VDD plane
W8	296	P	VSS plane
W9	297	I	testClkIn_h
W10	298	I	testClkIn_l
W11	299	P	VDD plane
W12	300	I	clkIn_h
W13	301	I	clkIn_l
W14	302	P	VSS plane
W15	303	P	VDD plane

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
W16	304	P	VSS plane
W17	305	P	VDD plane
W18	306	P	VSS plane
W19	307	P	VDD plane
W20	308	P	VSS plane
W21	309	I	tagadrP_h
W22	310	I	tagadr_h 32
W23	311	I	tagadr_h 31
W24	312	I	tagadr_h 30
Y1	313	B	data_h 8
Y2	314	B	data_h 71
Y3	315	B	data_h 7
Y4	316	B	data_h 68
Y5	317	P	VSS plane
Y6	318	P	VDD plane
Y7	319	P	VSS plane
Y8	320	P	VDD plane
Y9	321	P	VSS plane
Y10	322	P	VDD plane
Y11	323	P	VSS plane
Y12	324	P	VDD plane
Y13	325	P	VSS plane
Y14	326	P	VDD plane
Y15	327	P	VSS plane
Y16	328	P	VDD plane
Y17	329	P	VSS plane
Y18	330	P	VDD plane
Y19	331	P	VSS plane
Y20	332	P	VDD plane

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
Y21	333	B	adr_h 8
Y22	334	B	adr_h 5
Y23	335	P	VDD plane
Y24	336	I	tagadr_h 33
AA1	337	B	check_h 20
AA2	338	P	VDD plane
AA3	339	B	data_h 5
AA4	340	B	data_h 66
AA5	341	B	data_h 0
AA6	342	I	iAdr_h 6
AA7	343	I	iAdr_h 10
AA8	344	I	vRef
AA9	345	O	sysClkOut2_h
AA10	346	O	sysClkOut2_l
AA11	347	N	spare 6
AA12	348	O	sysClkOut1_h
AA13	349	O	sysClkOut1_l
AA14	350	I	cont_l
AA15	351	I	irq_h 5
AA16	352	N	spare 8
AA17	353	B	adr_h 31
AA18	354	B	adr_h 27
AA19	355	B	adr_h 24
AA20	356	B	adr_h 17
AA21	357	B	adr_h 15
AA22	358	B	adr_h 11
AA23	359	B	adr_h 7
AA24	360	B	adr_h 6
AB1	361	B	data_h 70

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
AB2	362	B	data_h 69
AB3	363	B	data_h 67
AB4	364	B	data_h 2
AB5	365	B	data_h 64
AB6	366	I	iAdr_h 7
AB7	367	I	iAdr_h 12
AB8	368	I	reset_l
AB9	369	I	sRomD_h
AB10	370	O	sRomOE_l
AB11	371	O	cpuClkOut_h
AB12	372	I	dcOk_h
AB13	373	I	triState_l
AB14	374	I	icMode_h 0
AB15	375	I	irq_h 4
AB16	376	I	perf_cnt_h 0
AB17	377	B	adr_h 32
AB18	378	B	adr_h 28
AB19	379	B	adr_h 25
AB20	380	B	adr_h 21
AB21	381	B	adr_h 18
AB22	382	B	adr_h 14
AB23	383	P	VSS plane
AB24	384	B	adr_h 9
AC1	385	B	data_h 6
AC2	386	P	VSS plane
AC3	387	P	VDD plane
AC4	388	B	data_h 65
AC5	389	P	VSS plane
AC6	390	I	iAdr_h 8

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
AC7	391	P	VDD plane
AC8	392	I	iAdr_h 11
AC9	393	P	VSS plane
AC10	394	O	sRomClk_h
AC11	395	P	VDD plane
AC12	396	N	spare 5
AC13	397	P	VSS plane
AC14	398	I	irq_h 2
AC15	399	P	VDD plane
AC16	400	I	perf_cnt_h 1
AC17	401	P	VSS plane
AC18	402	B	adr_h 29
AC19	403	P	VDD plane
AC20	404	B	adr_h 22
AC21	405	P	VSS plane
AC22	406	B	adr_h 16
AC23	407	P	VDD plane
AC24	408	B	adr_h 10
AD2	409	B	data_h 4
AD3	410	B	data_h 3
AD4	411	B	data_h 1
AD5	412	I	iAdr_h 5
AD6	413	I	iAdr_h 9
AD7	414	N	spare 1
AD8	415	I	eclOut_h
AD9	416	I	dInvReq_h
AD10	417	N	spare 2
AD11	418	N	spare 4
AD12	419	I	icMode_h 1

Table 8–1 (Cont.): 21064 Pin List

PGA Loc.	PIN No.	Type	Name
AD13	420	I	irq_h 0
AD14	421	I	irq_h 1
AD15	422	I	irq_h 3
AD16	423	N	spare 7
AD17	424	B	adr_h 33
AD18	425	B	adr_h 30
AD19	426	B	adr_h 26
AD20	427	B	adr_h 23
AD21	428	B	adr_h 20
AD22	429	B	adr_h 19
AD23	430	B	adr_h 13
AD24	431	B	adr_h 12

Introduction to Designing a System with the DECchip ™ 21064 Microprocessor

Revision/Update Information: Revision 1.0

Digital Equipment Corporation
Maynard, Massachusetts

April, 1992

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation 1992.

All Rights Reserved.
Printed in U.S.A.

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

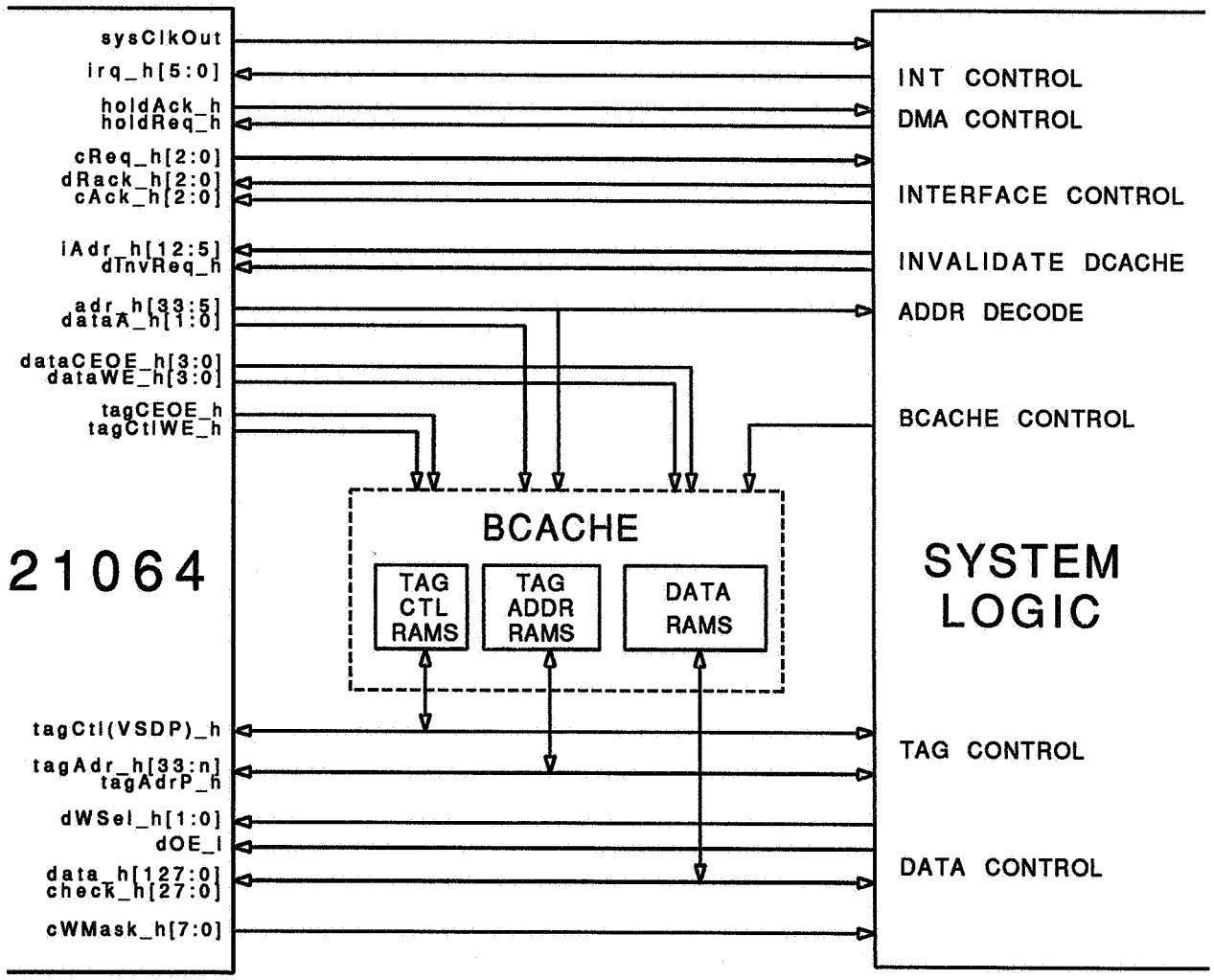
DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

This document was prepared using VAX DOCUMENT, Version 2.0.

1 Introduction

This application note provides a basic description of how to integrate the DECchip ™ 21064 microprocessor chip into a module or system. It describes how the processor reacts to the chip reset condition, and explains how to connect and control the chip interface signals. The 21064 chip has been designed to allow maximum flexibility while at the same time providing the ability to easily create a computing system with generally available module parts.

Figure 1: 21064 Pin Bus



This document is not meant to give every detail about interfacing to the chip. Rather, what is provided here is enough information for a design engineer to understand what is involved in creating a 21064-based system. It should allow the designer to quickly determine how 21064 system design compares with other design tasks.

Examples are used throughout the text to clarify meaning, but this is not intended to imply that what is described is the only way to use the chip. An attempt has been to describe real, usable circuits and techniques, but the chip is flexible and the designer is encouraged to investigate other implementations. A preliminary data sheet is available for the 21064 microprocessor that describes the details and additional features of the chip.

2 General concepts

Some important design concepts are common to many 21064-based system designs, and they are discussed in this section. The chip pin bus is flexible and mandates few design rules, leaving open a wide range of prospective systems. Figure 2 is a diagram of the 21064 pin bus, showing the major signal groups.

A system designed with the chip can be divided into three major sections. There is the 21064 processor itself, the system control logic, and the external backup cache (Bcache) between them (the Bcache is optional, though most systems will see a performance improvement if it is included). The chip pin bus provides address and control signals, and transfers data through a 128-bit bidirectional data bus. The preliminary data sheet describes each of the signals Figure 1 in detail.

The processor controls the Bcache when its initial tag probe finds that the information is valid and unshared. The Bcache access is under control of the CPU, and the external system logic is not involved. When the CPU does a Bcache probe and misses, or when a lock-associated command is invoked, the processor starts an external cycle.

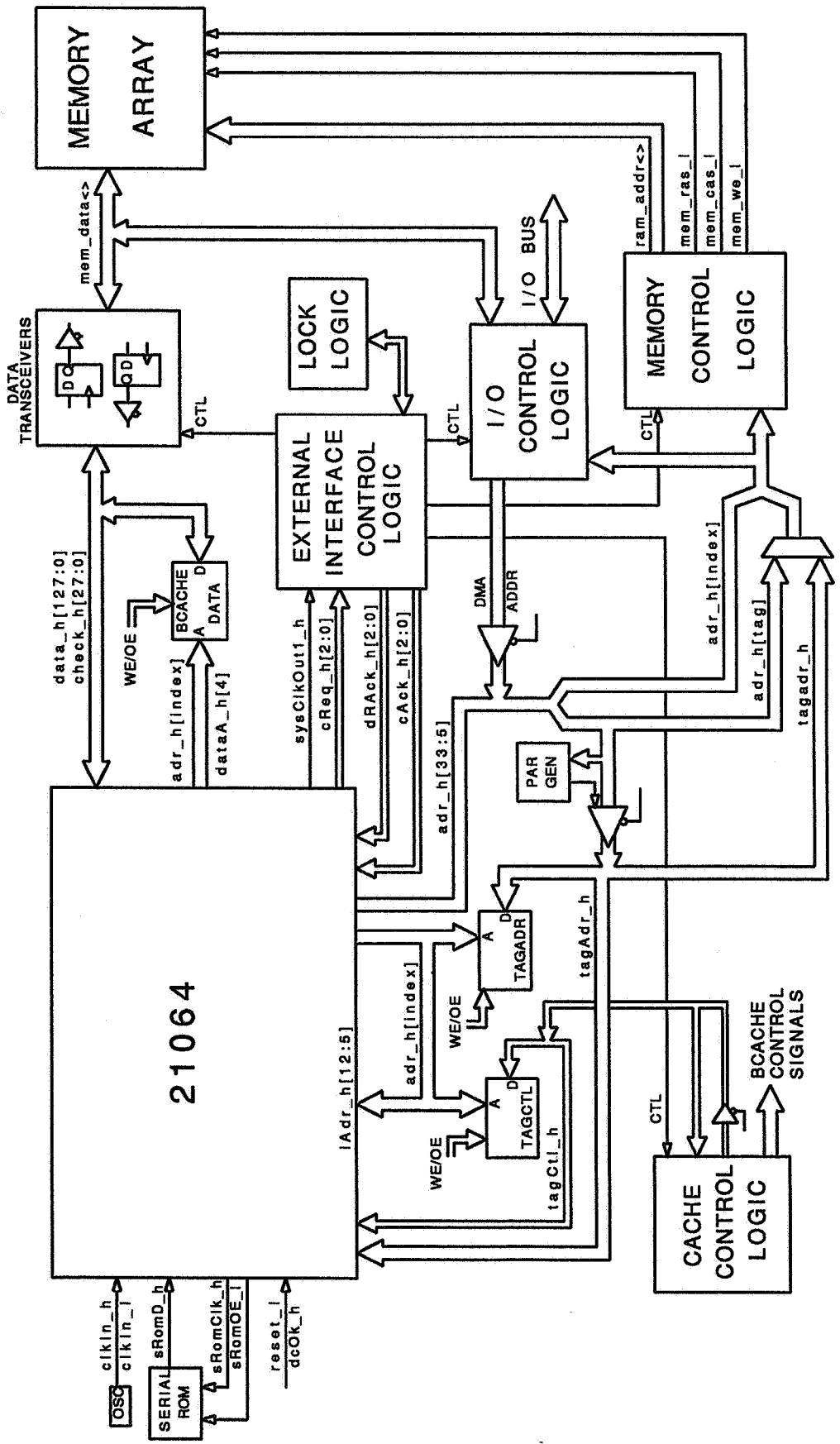
During the external cycle, the Bcache is under control of the system logic. The system logic either returns the data to the processor, or accepts the data from the processor (depending upon the cycle type), and acknowledges the cycle to give control back to the CPU. If the cycle necessitates a Bcache fill, it is up to the system logic to load the data into the Bcache RAMs, the upper address bits (with good parity) into the tag address RAMs, and the proper valid and parity bits into the tag control RAMs.

To help the design engineer create high performance systems more easily, the Bcache is controlled by the 21064 pin bus during probes that hit. This allows off-the-shelf SRAMs to be connected to the chip without a lot of extra components. The Bcache interface signals are programmable through an internal processor register (IPR), so that the Bcache size, access, and write timing can be set with complete flexibility without affecting the internal CPU clock speed.

That is, the 21064 can be running at its nominal 6.6ns internal cycle time, but the Bcache can run slower if required without slowing down the internal timing. There are two internal caches in the 21064 chip: an I-stream read-only cache (Icache) and a D-stream write-through cache (Dcache). The speed of the Bcache does not affect the internal caches, which use the internal clock.

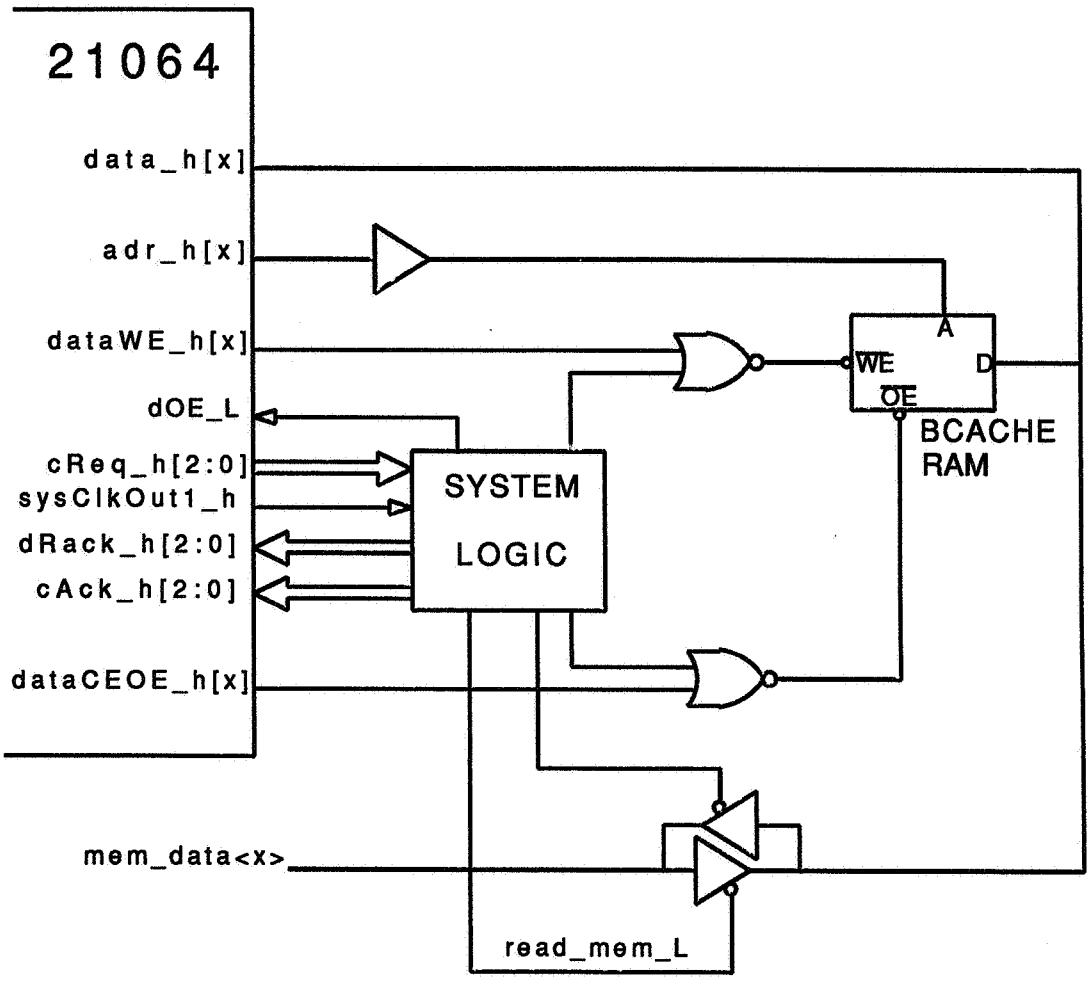
Figure 2 is a block of a system that can be created using the 21064 microprocessor. The major sections are shown, along with many of the buses that would run through such a system. In the center of the diagram is the external interface control, which directs the other system logic subsections that interface to memory, I/O, Bcache, etc.

Figure 2: 21064-Based System Block Diagram



The Bcache, if it is included in a system, can be as small as 128KB or as large as 8MB. The size is under program control. The **adr_h[33:5]** bus in Figure 2 is shown partitioned into an [index] field and a [tag] field. The size of each field depends upon the Bcache size. The smallest Bcache (128KB) uses **adr_h[16:5]** to index into the the cache block, and the tag field would be **adr_h[33:17]**. Only those bits that are actually needed for the amount of potentially cachable system main memory need to be stored in the Bcache tag, although the 21064 uses all the relevant tag address bits for that Bcache size on its tag compare. A larger Bcache uses more index bits and fewer tag address bits.

Figure 3: Bcache Control Logic



On an external request (read or write), the 21064 sends out the address and cycle type (and data for a write cycle), then waits until the system logic sends back the acknowledgment handshake that the cycle is complete. On a read request cycle, each data word is tagged as it comes back by the system logic with information about whether the data should be checked for ECC (or parity, depending upon which mode of operation has been selected for

the chip), and whether it should be cached inside the chip. On a write request, the system logic merely notifies the chip that the write has been accepted for processing.

The Bcache is shared between the 21064 and the system logic. Although the processor directly manipulates the Bcache for read and write hits, it is up to the system logic to:

- Fill the Bcache with memory data
- Load the tag address and tag address parity
- Load tag control bits and parity on fills (valid and non-dirty)
- Write data back to memory when necessary
- Probe the Bcache for lock/unlock transactions
- Probe and control the Bcache for DMA transactions

The Bcache control signals are thus under potential control of the 21064 or the system logic. When the CPU chip determines that an external cycle is necessary, it drives the Bcache control signals to false. This allows the system logic to read and write the Bcache RAMs. Figure 3 shows the expected configuration for the Bcache. The figure shows a data line, but the tag address and control lines are expected to be connected similarly.

The signal **mem_data** in Figure 3 is a bidirectional memory data bus that connects to the main storage. When it is necessary to load the contents of memory into the Bcache, the system logic will drive the memory bus control signals such that a read cycle is performed. In this example, the signal **read_mem_L** is being used to drive the Bcache (and 21064) data bus. The system logic will properly drive the Bcache RAM write enable signal, and once the data is stable on the **data_h[x]** bus, it will be strobed into the Bcache.

When the Bcache contents need to be written back to memory, the system logic will control the RAM output enable signal to access the Bcache data. The signal **read_mem_L** will now be de-asserted, and the memory control signals will also properly tristate the **mem_data** bus so that the data can be written to the memory storage elements. The system logic must properly assert the 21064 signal **dOE_l** so that the CPU will drive the **data_h[x]** lines.

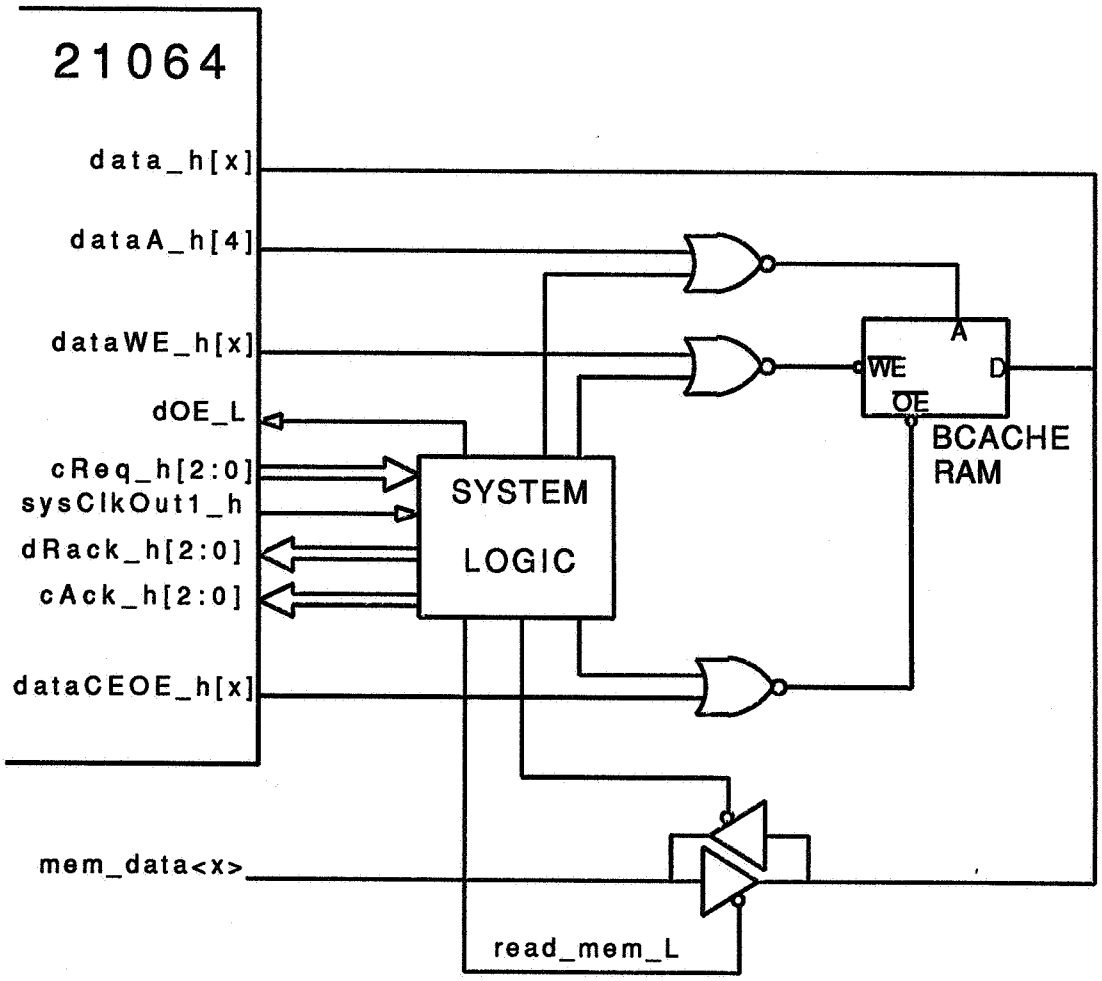
The Bcache consists of 32-byte blocks or larger. As such, the 21064 supplies address bits [33:5] to select which Bcache block. The CPU data bus is 16 bytes wide, and thus each Bcache cycle requires two accesses. The CPU outputs the signal **dataA_h[4]** to control which 16-byte data half is being written to or read from. Figure 4 shows the expected configuration for the lower address bit. As with the chip output enable and write pulse, the lower Bcache address bit is under control of either the 21064 or the system logic. When the CPU is in external system logic mode, it drives the **dataA_h[4]** signal low (along with the other Bcache control signals).

This application note will later go though some general cycle types, including timing diagrams to better explain how a 21064-based system functions.

3 Basic 21064 Power, Input Level, and Clock Issues

The preliminary data sheet describes how to power and clock the 21064 in detail. This section provides an overview of these issues, and some example circuits that can be used.

Figure 4: Lower Bcache Address



3.1 Power Supply and Input Levels

The 21064 is powered from a +3.3V supply (+/- 5%), but will drive and accept CMOS/TTL-compatible levels once the chip has been properly stabilized. It is *mandatory* that no input or bidirectional pin be allowed to rise above 4.0V until the 3.3V power to the chip is stable. Failure to follow this rule will damage the chip.

This rule does *not* imply that power supply sequencing must be used. It only means that any other module part that can drive the input pins must be kept in tristate mode until the 21064 has stable power. So, for example, a *dcOK* signal can be used to prevent components such as SRAMs, MUXes, and buffers from driving the chip.

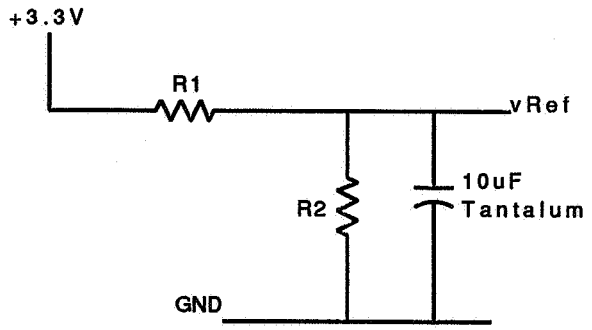
3.2 Input Level Sensing

The 21064 uses a reference input pin **vRef** to supply the threshold level for all chip inputs except:

- clkIn_h,_l
- testclkIn_h,_l
- tagOk_h,_l
- dcOk_h
- eclOut_h
- tristate_l
- cont_l

These pins should never be driven above the 21064 power supply. Since the nominal voltage to the chip is 3.3V, care must be taken if any of the signals above are generated from logic that has a 5V supply. Note especially that **dcOk_h** is one of the signals that must never be driven above the nominal 3.3V level, since it is likely that it will be generated from a higher voltage.

Figure 5: Input Reference Voltage Circuit



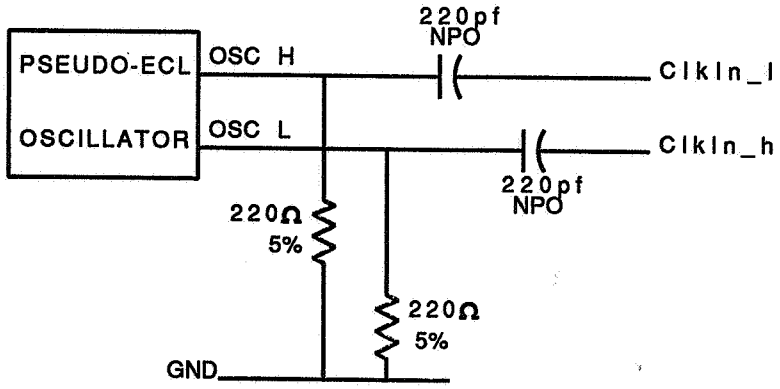
vRef should be connected to a stable 1.4V (+/-10%) source. Figure 5 can be used to supply this voltage level. The resistors R1 and R2 in the figure should be chosen so that they form a voltage divider that supplies the 1.4V level to **vRef**. **vRef** has a large capacitance on it inside the chip, and there is an RC delay between its pin and the other input buffers. Therefore, **dcOk_h** should not be asserted until there has been enough time for the **vRef** input to stabilize. Consult the preliminary data sheet for more details concerning the assertion of **dcOk_h**.

Note that **reset_l** *is* one of the input pins that uses **vRef** for its threshold level, so it cannot be relied upon until **vRef** is stable. **dcOk_h** being false (that is, low) keeps the chip in reset mode.

3.3 Input Clocks

The 21064 expects differential clock signals between 0.6V and 3.0V for the **clkIn_h,_l** inputs. A reversed can pseudo-ECL oscillator with pulldowns can be AC-coupled to the clock inputs for this purpose. Using a pseudo-ECL oscillator means you don't have to design a special ECL power supply to clock the chip. Figure 6 is an example of a working circuit. Note that the series capacitor should use an NPO dielectric.

Figure 6: Input Clock Circuit



Up to 200MHz (translating to a 10ns internal CPU clock cycle), a lower-cost 10K-series oscillator will work fine. Above that speed, a 100K-series oscillator should be used.

Due to internal chip circuitry, the test clock input signals **testClkIn_h,_l** should be pulled to the appropriate level using small resistors (100 ohms maximum). **testClkIn_h** should be pulled high (that is, to 3.3V through a small resistor) and **testClkIn_l** should be pulled low (to ground).

3.4 Unused Inputs

There are several inputs that are not used in a 21064-based system, but must be tied off either high or low. The following inputs should be pulled to 3.3V through a resistor:

- tagOk_h (unless using the tagOk function)
- tristate_l
- cont_l
- perfCntIn_h[1:0]

The following inputs should be pulled to ground:

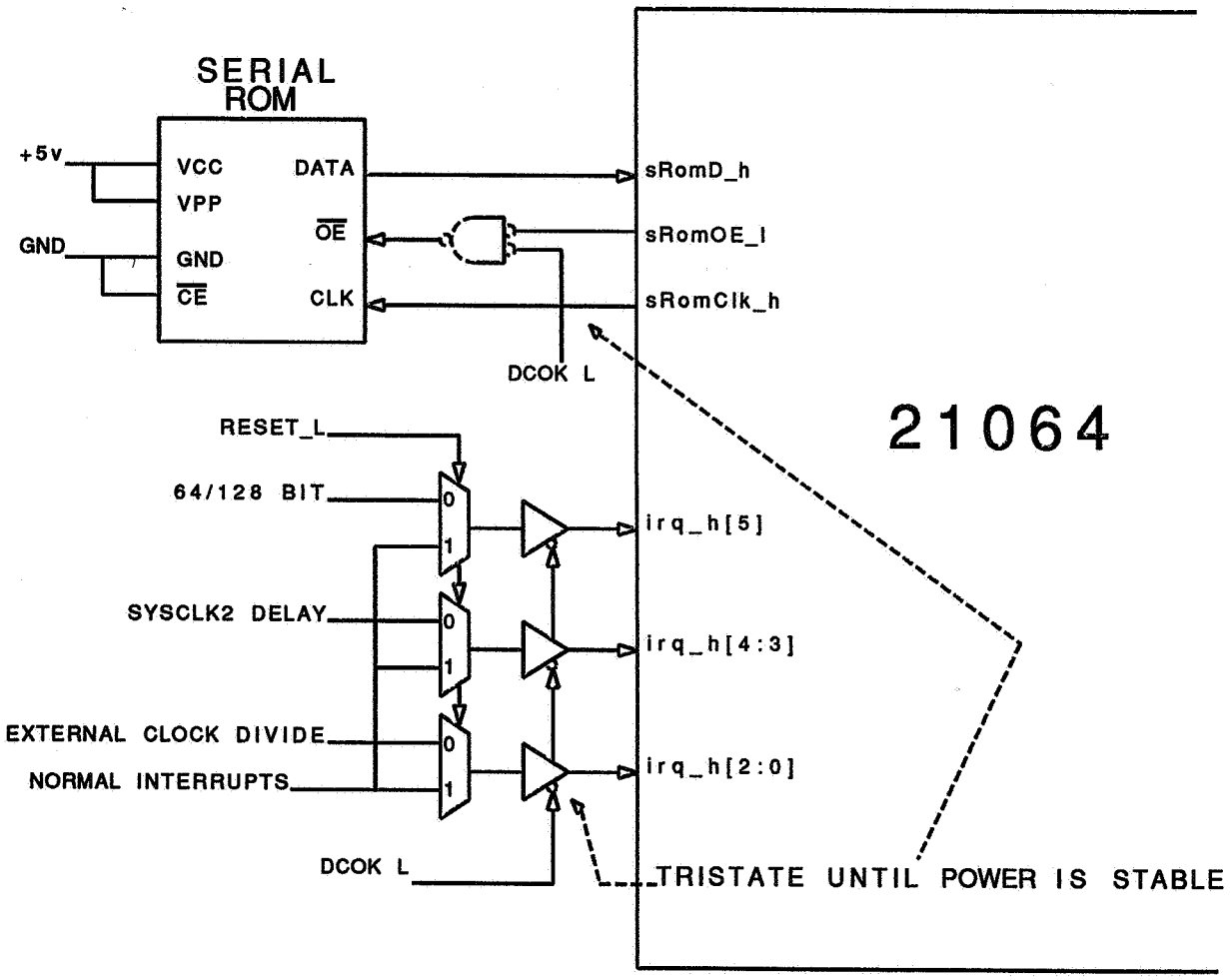
- tagOk_l (unless using the tagOk function)
- dWSel[0] (unless in 64-bit data bus mode)
- eclOut_h
- icMode_h[1:0]

The **tagOk_h,l** signals are used to stall the 21064 so that the Bcache can be controlled by the system logic. They are optimized for very high performance systems, and are not discussed in this document. The preliminary data sheet provides more details about the signals and their use. This application note discusses the simpler *holdReq_h* method for the system logic to take control of the Bcache (see Section 8.

4 Booting the 21064

The 21064 uses a flexible method to bootstrap the processor. Instead of always jumping to a fixed I/O address upon reset, the chip can load its initial I-stream from a compact serial ROM (SROM). As well, the configuration of the external interface is programmable by setting up certain input pins at reset time. Figure 7 shows how the serial ROM and the configuration inputs are used at reset time.

Figure 7: Serial ROM and Programmable Clock Inputs



While the 21064 is in reset mode, the interrupt input lines **irq_h[5:0]** are inspected to determine how the chip should configure the external interface logic. There are three configurable areas:

1. The 21064 can accommodate either a high-performance 128-bit external data bus or a lower-cost 64-bit data bus. **irq_h[5]** determines which of the two is selected, and is asserted high to choose the 128-bit mode. This application note describes the 21064 in 128-bit mode, but the preliminary data sheet provides more information about the differences.
2. The external interface runs synchronously to the external system clock, **sysClkOut1_h**. This external clock is generated from the internal clock, which can be divided by any value between 2 and 8 to form **sysClkOut1_h**. So, for example, the 21064 chip running at its nominal 6.6ns internal clock cycle time can be divided by 4 to allow an external interface to run at 26.4ns. **irq_h[2:0]** select the external interface division factor. Table 1 is a chart of the clock divisor decode.
3. The external interface logic is supplied two differential clocks from the 21064, **sysClkOut1_h,_l** and **sysClkOut2_h,_l**. Each external clock runs at the external cycle time selected above. **sysClkOut2** can also be delayed from **sysClkOut1** by a programmable value selected from **irq_h[4:3]**. The second clock can be delayed from 0 to 3 internal CPU clocks based upon this selection. Table 2 shows the delay times possible and their decode meaning.

Table 1: System Clock Divisor

irq_h[2]	irq_h[1]	irq_h[0]	Ratio
0	0	0	2
0	0	1	3
0	1	0	4
0	1	1	5
1	0	0	6
1	0	1	7
1	1	0	8
1	1	1	8

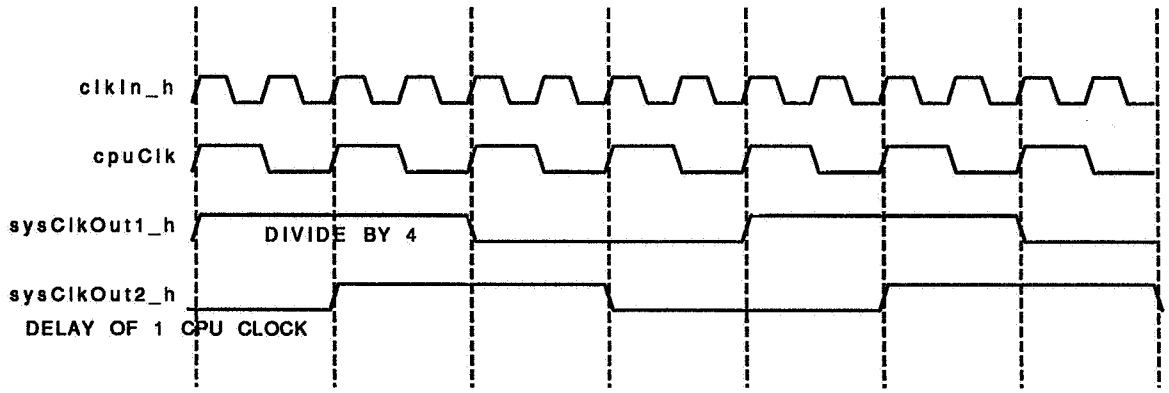
Table 2: System Clock Delay

irq_h[4]	irq_h[3]	Delay
0	0	0
0	1	1
1	0	2
1	1	3

Figure 8 shows how the clock configuration works. The input clock that is provided to the 21064 chip is divided by 2 in order to create the internal CPU clock. The CPU clock is the reference to all the other clocks that the chip outputs. In the example, the clock divisor is 4, so the system output clocks run at 1/4 of the internal CPU clock time. The figure shows that

sysClkOut2 has been delayed by 1 CPU clock from **sysClkOut1**. Since the external output clocks are differential, a two-phase clock is also available by using **sysClkOut1_h,_l**.

Figure 8: Example of 21064 Clock Configuration



When the **reset_l** signal de-asserts, the serial ROM is loaded into the processor Icache. The CPU controls the output enable and the clock for the ROM, and accepts the bit serial data. The preliminary data sheet provides details about the timing of the SROM control signals. After the SROM data has been loaded into the Icache, the processor jumps to location 0, which will hit inside the Icache. The SROM code is expected to perform chip and system initialization, preparing the system for external operation.

After the SROM code has been executed, it is assumed that the external interface is ready to supply I-stream data to the 21064 processor. The Bcache can be on or off at this point (in fact, there is no need to even have a Bcache if the user has no performance reason to include it). A general system might include a more complete boot/diagnostic ROM (BDROM) after the SROM has done its job.

Once the 21064 is executing in I-stream mode from an external interface, it expects full 32-byte fills. The normal data path of the 21064 is 128 bits (16 bytes), so two complete fill cycles are necessary to provide the 32 bytes of data. The BDROM code can be loaded and executed in several ways, though the suggested method is to move the BDROM code into RAM memory, then execute it from there. This can be easily handled by the serial ROM, which can read the BDROM byte by byte, pack it into appropriate memory words, move it into main memory, then jump to it in RAM.

5 Cache/memory Interface Details

The Bcache subsystem is carefully integrated into the 21064. It is expected that the Bcache SRAMs can be directly controlled by the 21064 pin bus, and that the Bcache data lines are connected to the 21064 data bus, as shown in Figure 2.

The rest of this description assumes that a Bcache does exist and is enabled. The case where a Bcache is not part of the data path is much simpler, and this same document can be used to understand the design of such a system by merely ignoring those sections dealing with the Bcache.

The Bcache is organized into 32-byte blocks or larger, with parity or ECC on 4-byte (32-bit) segments (no error detection is also an option). When the Bcache is enabled, the 21064 will generally probe it for each memory access (lock-related cycles are an exception). The tag and control SRAMs will first be enabled at the appropriate address, and if the probe finds a valid match the cycle will be finished without performing a main memory read or write cycle. The first 128-bit (16-byte) data segment will be read at the same time as the Bcache tag probe, and will be ready if the probe is successful. The 21064 will then read the second 128-bit segment. If the internal cache is enabled, the data is saved inside the chip. The preliminary data sheet provides a timing diagram of the 21064-controlled Bcache access cycles.

The Bcache is best utilized in writeback mode, which means that both reads and writes are normally serviced from the Bcache without external logic intervention. This implies that the Bcache has the only valid copy of a data block after it's been modified. The 21064 will manipulate the Bcache DIRTY bit to signify that the block has been written since it was initially read from memory. There is a method that the system logic can use to force non-writeback behavior, but its use is beyond the scope of this document. The preliminary data sheet discusses the **SHARED** Bcache bit in more detail.

5.1 Bcache Timing for 21064 Access

The Bcache timing is under complete control of the user through the BIU_CTL internal processor register (IPR). Figure 9 shows the layout of this register, which will normally be set up as part of the chip initialization code. The number of internal CPU cycles to allocate for Bcache reads and writes can be specified, along with the exact representation of where the Bcache write pulse will be asserted for Bcache writes.

Figure 9: BIU_CTL Internal Processor Register

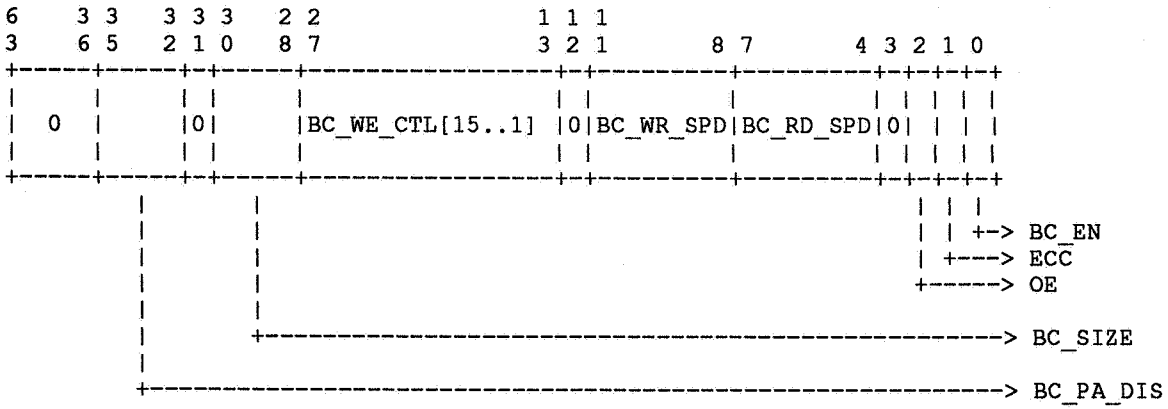


Table 3: BIU Control Register

Field	Type	Description
BC_EN	WO,0	Bcache enable. When clear, this bit disables the Bcache. When the Bcache is disabled the 21064 does not probe the Bcache tag store for read and write references; it launches a request on cReq_h immediately.
ECC	WO	When this bit is set, the 21064 generates/expects ECC on the check_h pins. When this bit is clear the processor chip generates/expects parity on four of the check_h pins.
OE	WO,0	When this bit is set, the 21064 does not assert its chip enable pins during RAM write cycles, thus enabling these pins to be connected to the output enable pins of the cache RAMs.
BC_RD_SPD	WO	Bcache read speed. This field indicates to the BIU the read access time of the RAMs used to implement the off-chip Bcache, measured in CPU cycles. It should be written with a value equal to one less the read access time of the Bcache RAMs. Access times for reads must be in the range 16..3 CPU cycles, which means the values for the BC_RD_SPD field are in the range of 15..2. BC_RD_SPD are not initialized on reset and must be explicitly written before enabling the Bcache.
BC_WR_SPD	WO	Bcache write speed. This field indicates to the BIU the write cycle time of the RAMs used to implement the off-chip Bcache, measured in CPU cycles. It should be written with a value equal to one less the write cycle time of the Bcache RAMs. Access times for writes must be in the range 16..2 CPU cycles, which means the values for the BC_WR_SPD field are in the range of 15..1. BC_WR_SPD are not initialized on reset and must be explicitly written before enabling the Bcache.
BC_WE_CTL	WO	Bcache write enable control. This field is used to control the timing of the write enable and chip enable pins during writes into the data and tag control RAMs. It consists of 15 bits, where each bit determines the value placed on the write enable and chip enable pins during a given CPU cycle of the RAM write access. When a given bit of BC_WE_CTL is set, the write enable and chip enable pins are asserted during the corresponding CPU cycle of the RAM access. BC_WE_CTL[0] (bit 13 in BIU_CTL) corresponds to the second cycle of the write access, BC_WE_CTL[1] (bit 14 in BIU_CTL) to the third CPU cycle, and so on. The write enable pins will never be asserted in the first CPU cycle of a RAM write access. Unused bits in the BC_WE_CTL field must be written with zeros. BC_WE_CTL is not initialized on reset and must be explicitly written before enabling the Bcache.
BC_SIZE	WO	This field is used to indicate the size of the Bcache. BC_SIZE is not initialized on reset and must be explicitly written before enabling the Bcache. See Table 4 for the encodings.

Table 3 (Cont.): BIU Control Register

Field	Type	Description
BC_PA_DIS	WO	This 4-bit field may be used to prevent the CPU chip from using the Bcache to service reads and writes based upon the quadrant of physical address space which they reference. The correspondence between this bit field and the physical address space is shown in Table 5. When a read or write reference is presented to the 21064 the values of BC_PA_DIS, BC_ENA and physical address bits [33:32] together determine whether to attempt to use the Bcache to satisfy the reference. If the Bcache is not to be used for a given reference the chip does not probe the tag store, and makes the appropriate system request immediately. The value of BC_PA_DIS has NO impact on which portions of the physical address space may be cached in the primary caches. System components control this via the dRack field of the pin bus. BC_PA_DIS are not initialized by reset.

Table 4: BC_SIZE

BC_SIZE	Size
0 0 0	128 Kbytes
0 0 1	256 Kbytes
0 1 0	512 Kbytes
0 1 1	1 Mbytes
1 0 0	2 Mbytes
1 0 1	4 Mbytes
1 1 0	8 Mbytes

Table 5: BC_PA_DIS

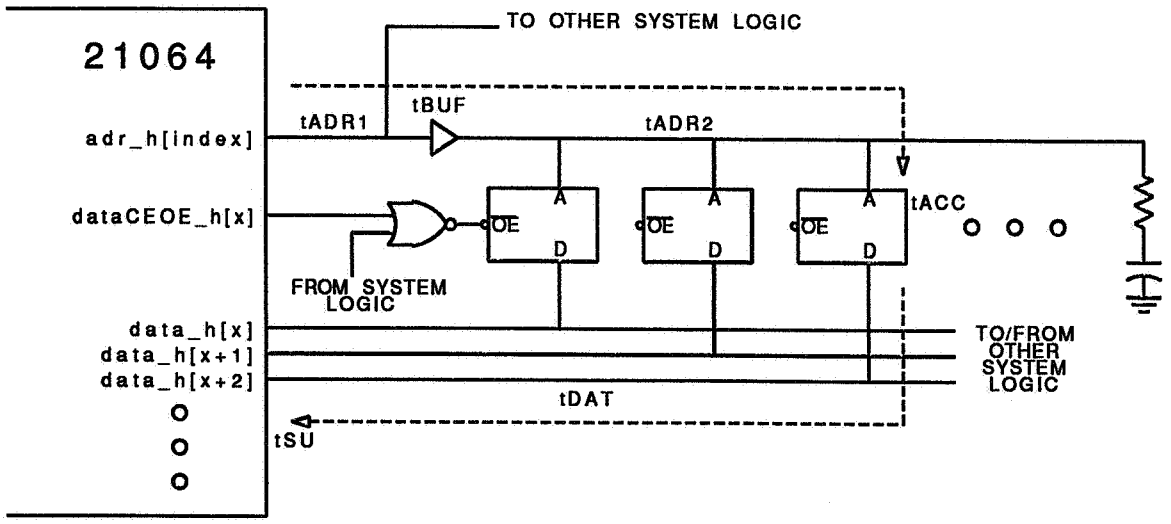
BIU_CTL bits	Physical Address
[32]	PA[33..32] = 0
[33]	PA[33..32] = 1
[34]	PA[33..32] = 2
[35]	PA[33..32] = 3

5.1.1 Bcache Read Cycle

For a Bcache read cycle, the access/cycle time is determined by adding up the complete address or control path from the 21064 pin bus until the data is valid at the 21064 data bus pins. There is a 5ns setup requirement inside the 21064 on data reads, and this must also be considered. A system designed with the 21064 must provide access to the Bcache address and control signals from the module logic, so there is a NOR-type gate in the path. Furthermore, the 21064 output buffers are characterized driving a 40pf load, so any large

fanout must be accomplished without exceeding this value. This usually means that buffers are added to the address and control paths.

Figure 10: Bcache Access Path for 21064



An example of a Bcache read access time calculation is provided here to clarify the steps. Figure 10 shows the general circuit assumed for this example. The address path drive signals will normally be treated as transmission lines in a real high-performance Bcache, so that is how they will be shown here. The termination scheme indicated in the figure assumes that your address buffer can drive a low impedance line to a proper level on the incident wave. If your driver cannot do this, then series termination should be used, with the implied increase in delay time due to the necessary reflection for a proper signal level. We will assume that the address buffer in the example has a specified propagation delay of 5ns. One of the address lines is assumed to be a fast, high-drive capability NOR-gate, and for our purposes it will be treated like the address buffer.

Many devices specify the maximum propagation delay with only one output switching, and in the case of an address buffer all the outputs might switch simultaneously. To account for this, extra buffer delay should be added to the assumed propagation delay through the device. For this example, we will assume that the 5ns buffer delay takes this into account.

All the calculations shown here are based upon the assumptions stated. The system or board designer is responsible for analyzing any particular implementation, and determining the correct delays and signal integrity issues. The purposes of this example are to show a general Bcache circuit that can be implemented with the 21064, and to explain how to program the IPR that controls the Bcache. Faster and slower systems can be built with the 21064 processor.

The SRAMs in our example have a specified access time of 20ns from address stable to data valid at their output pins. SRAM devices often have a faster specification from output enable to data valid, and it will be assumed that the *address* path, not the output enable path, is the critical one. The designer should ensure that this is true for any specific implementation.

The output enable path can be analyzed similarly to the address path. So the general components of delay for this calculation are:

```
tADR1  [delay from CPU to input of address buffer]
tBUF   [buffer gate delay]
tADR2  [address delay from buffer to SRAM inputs]
tACC   [SRAM access time from address valid to data valid]
tDAT   [data return path from SRAM to 21064 input pins]
tSU    [internal 21064 data setup time]
```

Figure 11: Timing Diagram for Bcache Read Access

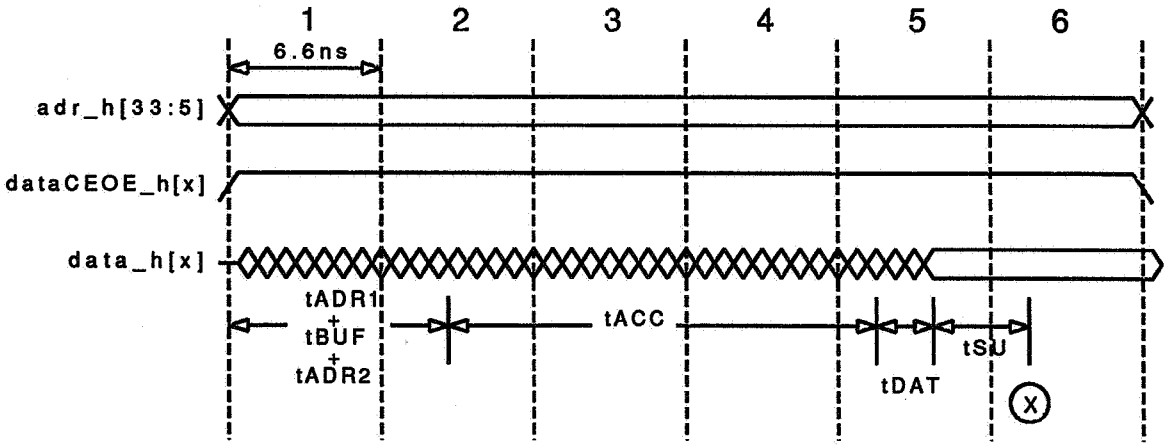


Figure 11 is a timing diagram showing the 21064 signals and their delay components. The valid data cannot be sampled until the point labeled "X" in the figure. The preliminary data sheet provides more detailed timing diagrams of the fast Bcache access path. The Bcache probe and each data read access would have the timing shown in Figure 11, and they are controlled by the same programmable BIU_CTL field.

The three unknown delay components are the address paths (tADR1, tADR2) and the data return path (tDAT). The data return path (tDAT) depends on the edge rate of the SRAM output, the length of the data line, and the other loads that are connected to the data line. As such, it is impossible to specify a "normal" delay time. For this exercise, it will be assumed to be 2ns.

The address delay path from the 21064 address output to the buffer (tADR1) is similar to the data path. It is unlikely to be a classical transmission line, due to the line length in relation to the edge rate of the 21064 output. However, there will likely be other loads on the address line, and the etch itself will cause a delay of around 160ps to 200ps per inch. For this example, tADR1 will be specified to be 2ns.

The path tADR2 needs a more classical transmission line analysis, since the buffers will have a fast switching time in relation to the line length. Even if the address drivers can switch the line to a proper level on the incident wave, the wave will propagate along the transmission line more slowly than if it was unloaded. Each SRAM will contribute some capacitance to the line, which will slow the wave down according to the formula:

```
tPL = tPD * SQRT(1+Ca/Co)
```

The term tPL is the loaded propagation delay per unit length, tPD is the propagation delay per unit length of the unloaded line, Ca is the added capacitance per unit length due to the SRAM inputs, and Co is the unloaded transmission line capacitance per unit length. It will be assumed for this example that it will take the wave 2ns to reach the last SRAM address input, where there will be no reflection. If the address driver cannot switch the line on the incident wave, a series termination scheme would be used instead, and the delay value would be higher.

So, the full trip from address valid at the 21064 output pin to data valid at the 21064 input pin (plus data setup) is:

```
 2ns   tADR1
 5ns   tBUF
 2ns   tADR2
20ns   tACC
 2ns   tDAT
 5ns   tSU
-------
36ns
```

The numbers above are only for this example. If the designer uses different buffers, or splits the address drivers differently, or uses drivers that cannot switch the low impedance line on the incident wave, the analysis would change accordingly. We will assume that the 21064 is using an internal cycle time of 6.6ns, which means that the chip must allocate 6 cycles for the Bcache read given the conditions specified. This is programmed into the BIU_CTL register by setting the BC_RD_SPD field to 5, since the actual cycle count is one more than the one specified in the register. This value will work for any round trip delay that is less than or equal to 39.6ns.

It should be noted that using SRAMs with an access time of 17ns would reduce the number of internal CPU cycles to 5, assuming that everything else remained constant.

5.1.2 Bcache Write Cycle

A fast CPU-activated Bcache write cycle can be analyzed similarly. The BC_WR_SPD field in the BIU_CTL register should be programmed so that the SRAM write cycle will finish, and the BC_WE_CTL field should place the write pulse so that the timing and width do not violate the SRAM specifications.

An example of this calculation is provided here. Figure 12 shows the circuit that is assumed for the Bcache write path.

Figure 13 shows a timing diagram of the write path signals as viewed from the 21064. The preliminary data sheet provides a detailed timing diagram of a fast Bcache write access. The tag probe follows the timing for a fast Bcache read, and each write access follows the timing as shown in Figure 13. The write pulse cannot assert until point "X" in the figure, and it cannot de-assert until point "Y" in the figure. The cycle cannot end until point "Z" in the figure.

Figure 12: Cache Write Path for 21064

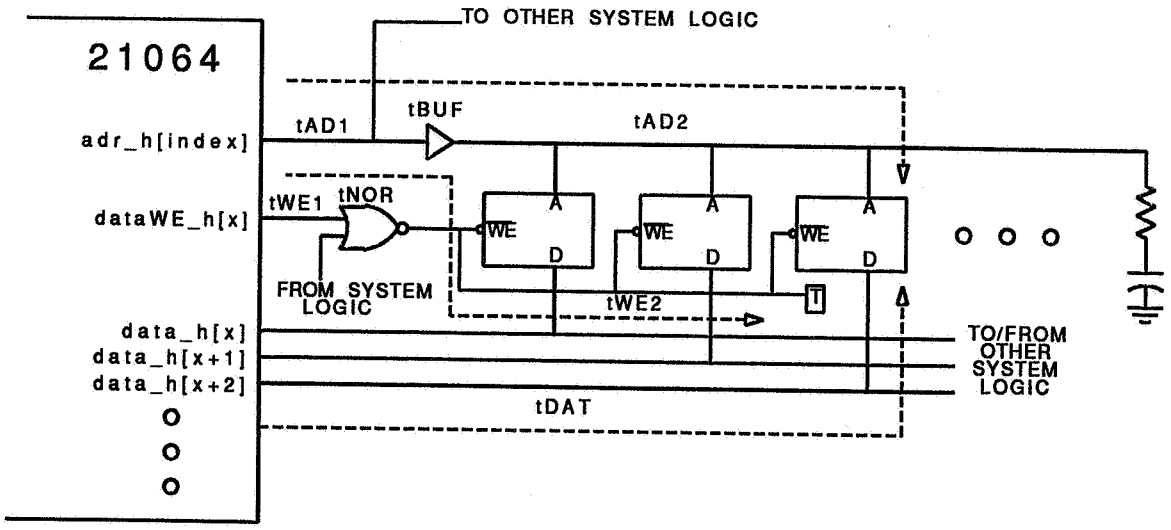
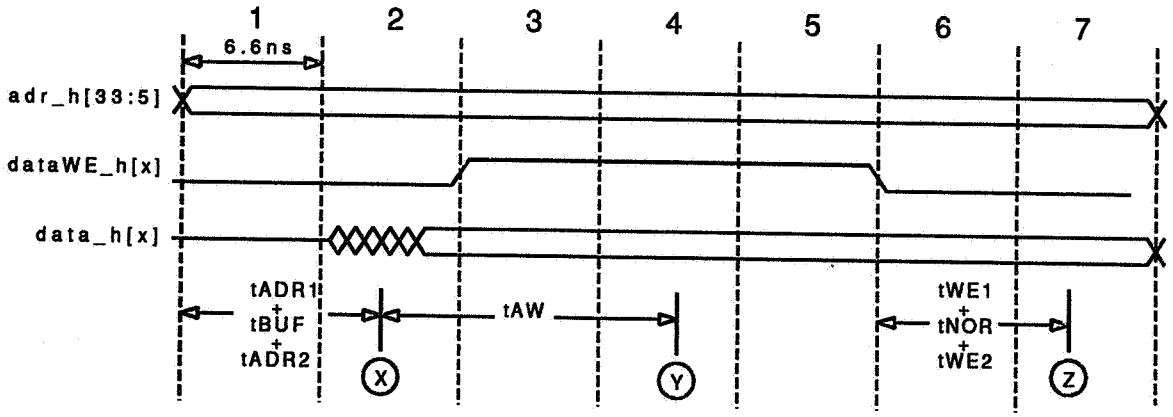


Figure 13: Timing Diagram for Bcache Write Access



It will be specified for this example that the minimum write pulse width for the SRAM (tWM) is 15ns. The 21064 can have 1.5ns of skew between the rising and falling edges of the pulse it will generate. Furthermore, although the rise and fall delays through the NOR gate in the figure should be close, some skew must be added to account for:

- Potential input threshold differences inside the SRAM
- Differences that result in a rise propagation delay that is different than the fall propagation delay

For the purposes of this example, we will add 2ns of skew between the rising and falling edges of the write pulse (1.5 for the 21064, and 0.5 for the logic and threshold differences). The following SRAM specifications will be used for this example:

```
tWC = 20ns  [Write cycle time]
tWP = 15ns  [Write pulse width]
tDW =  8ns  [Data setup time to write pulse de-assertion]
tDH =  0ns  [Data hold time from write pulse de-assertion]
tAW = 15ns  [Address setup time to write pulse de-assertion]
tWR =  0ns  [Address hold time from write pulse de-assertion]
tAS =  0ns  [Address setup time to write pulse assertion]
```

The above specifications are only a subset of the total device specifications for a real device, and are used to show the general technique used in determining how to program the BIU_CTL IPR. Designers should closely analyze their own systems, including the device support logic, etch paths, and RAM specifications, in order to determine exactly which paths are critical.

The first BIU_CTL field to calculate is the BC_WE_CTL, which determines where the write enable pulse will be asserted. The field is 15 bits wide, and each bit represents an internal CPU cycle that will assert the write enable pulse (starting with the second cycle, since the first cycle will never drive the write enable pulse).

The address delay calculation is similar to the read case, and it should be added to the address setup time as follows:

```
 2ns   tADR1
 5ns   tBUF
 2ns   tADR2
15ns   tAW for SRAM
-------
24ns
```

The earliest that the write pulse can be de-asserted is then 24ns from the start of the write cycle, based upon the address setup requirement.

There are two types of "data" that need setup and hold time for the write cycle. The actual Bcache data is the first type, and the tag control inputs (VALID, DIRTY, SHARED, and PARITY) are the second type. The 21064 drives the tag control inputs one CPU cycle later than the actual data, and we will assume that they are the critical path. The chip will provide stable data at most 2.9ns after the nominal edge that drives the data (in this case, tag control) lines. We will assume that the data will take 2ns to get to the SRAMs and be stable. If the CPU clock cycle is 6.6ns, then the earliest that the write pulse can de-assert is calculated as follows:

```
 6.6ns   [1 CPU clock cycle]
 2.9ns   [21064 data stable time]
 2.0ns   tDAT
 8.0ns   tDW for SRAM
--------
19.5ns
```

It would appear that in this example the address path is the critical one, and the write pulse cannot de-assert until 24ns after the start of the write cycle. The minimum pulse width is specified to be 15ns, which must be extended to (15+2=) 17ns to account for the pulse width skew in the 21064 and the external logic. At an internal 6.6ns CPU cycle time, 3 cycles must be used for the write pulse.

Since the earliest that the write pulse can de-assert is 24ns after the start of the write cycle, the latest that it can assert (in order to meet that de-assertion time) is (24-17=) 7ns after the cycle start. We have specified here that the write pulse cannot assert until the address is

stable (tAS), and this will put a bound on how early the write pulse is asserted. It was determined previously that the address will reach the last SRAM (tADR1+tBUF+tADR2=2+5+2=) 9ns after the start of the cycle. Since there is also 1.5ns of skew between the address signal and the write pulse signal coming from the 21064, the real minimum time is (9+1.5=) 10.5ns from the cycle start.

So, the earliest that the 21064 can assert the write pulse is 13.2ns into the Bcache write (that's the beginning of the 3rd CPU cycle). The write pulse should then start at the 3rd CPU cycle and extend until the end of the 5th cycle. The BC_WE_CTL field should be programmed to be 000000000001110. This means that the write pulse will remain asserted until (6.6ns*5=) 33ns into the Bcache write, which puts it after the 24ns limit previously calculated.

The other programmable field of interest in the BIU_CTL is the BC_WR_SPD field, which determines the entire write cycle time. The write pulse itself is de-asserted at the end of the 5th CPU cycle into the Bcache write in this example, which means it nominally de-asserts (6.6*5=) 33ns from the start of the cycle. It might be 1.5ns later than that due to 21064 output skew. There is also a NOR gate in the path (tNOR), and some wire travel time associated with the signal (tWE1 and tWE2).

There are three components of delay for the write enable pulse. The two write delay components (tWE1 and tWE2) might or might not be transmission lines. Figure 12 implies that tWE1 is not a transmission line and tWE2 is, with parallel termination. The module designer should analyze the particular implementation to see what the correct configuration should be, and if one of them is a transmission line it should be terminated appropriately (this analysis is similar to the address calculation in the previous section).

We will assume that tWE1 is 1ns, tNOR is 5ns, and tWE2 is 2ns for this example. So, the latest that the write pulse can de-assert at the last SRAM (and thus the earliest that the cycle can end) is:

```
33.0ns   [nominal write pulse de-assertion from start of write]
 1.5ns   [21064 skew from nominal edge]
 1.0ns   tWE1
 5.0ns   tNOR
 2.0ns   tWE2
-------
42.5ns
```

At a 6.6ns cycle time this translates to 7 cycles, so the value of 6 should be programmed into the BC_WR_SPD field (since this value is always 1 less than the actual write cycle time). The nominal write cycle speed will be 46.2ns for this example. As with the read cycle, it will be noted here that if the write enable pulse requirement was shorter (say 11ns rather than 15ns), the fast Bcache write could be reduced to 6 cycles.

5.2 Bcache Miss and External Request

An initial Bcache fill operation is executed when the 21064 attempts to read or write a block that misses in the Bcache (the write fill operation assumes a write-allocate Bcache policy). The miss can be caused for several reasons:

1. The Bcache block for that index is not valid
2. The Bcache block for that index is valid, but the tag misses

The first scenario above is the simplest, and will be discussed first. When a Bcache probe results in a miss, an external READ_BLOCK or WRITE_BLOCK operation is initiated by the 21064 external interface logic. The READ_BLOCK and WRITE_BLOCK external cycles are the most basic method of transferring data between the 21064 and the system, and are discussed in some detail in this document. The preliminary data sheet provides more details about other command types.

The command is initiated when the 21064 places the appropriate code on the **cReq_h[2:0]** lines during the rising edge of **sysClkOut1_h**. Timing for external cycles is synchronous to **sysClkOut1_h**, and all setup and hold times are referenced to the rising edge of this clock. The address, control, and data signals all change simultaneously with **sysClkOut1_h**, and therefore cannot be sampled on that same edge. In general this is only a concern for those lines that are used to determine if a cycle should begin, such as the request lines **cReq_h[2:0]** (the **holdAck_h** line is also in this category, as discussed later). A delayed version of **cReq_h[2:0]**, perhaps sampled by **sysClkOut2_h**, should be used to feed any state machines that run on **sysClkOut1_h** and use the request lines.

Figure 14: External Cycle

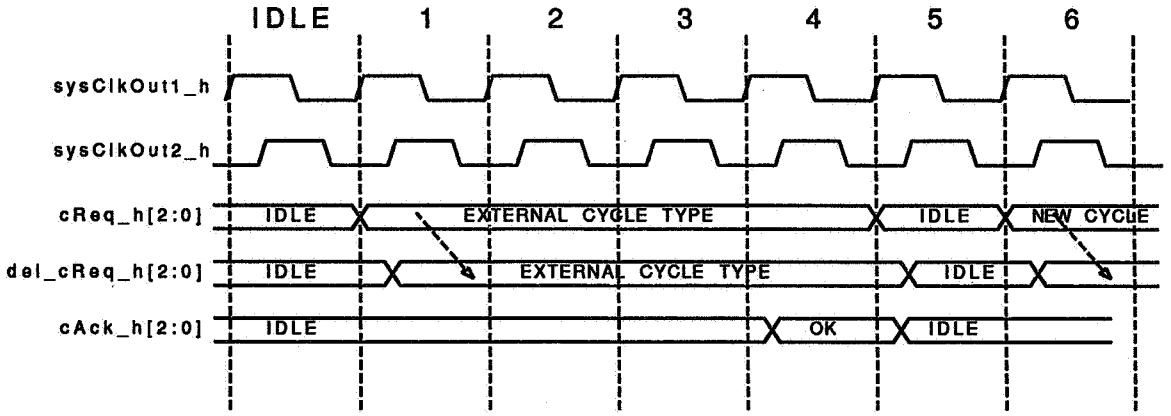


Figure 14 shows this relationship. The 21064 places the external cycle type on the **cReq_h[2:0]** lines at the start of cycle 1 in the figure. **sysClkOut2_h** is used to sample the request lines, and the system logic uses this delayed version to start its state machines at the start of cycle 2. After the external logic has performed the appropriate function, it changes the **cAck_h[2:0]** lines, which are sampled by the 21064 at the start of cycle 5. The CPU removes the request lines at that same time, and could start a Bcache access immediately (at the start of cycle 5). The earliest that the CPU can start another external cycle is one system clock cycle later, at the start of cycle 6 (as shown).

It is assumed here that the Bcache block is invalid, but the external logic would have no way to know that. So, the external logic must have some way to determine if the current Bcache block occupant needs to be written back to the main memory. One method to do this is to have the system logic perform its own Bcache tag probe. Only the VALID and DIRTY

bits need to be inspected, so the external logic probe does not have to wait the entire time necessary to compare the tag address field in the Bcache.

A critical path in this external logic probe is the SRAM output enable circuitry. The 21064 leaves the Bcache RAMs disabled after its own probe, and it's up to the external logic to drive the output enable again in order to inspect the VALID and DIRTY bits. One way to do this is to allow an early version of the **cReq_h[2:0]** signals to turn on the SRAM output enables by default, assuming that a probe will be necessary. For those cycles where the external logic later needs to write the Bcache, another logic path is necessary to turn the output enable back off. The de-assertion path is not time-critical, but does need to be implemented for cache fill operations.

Figure 15: Tag Control Probe Before External Cycle

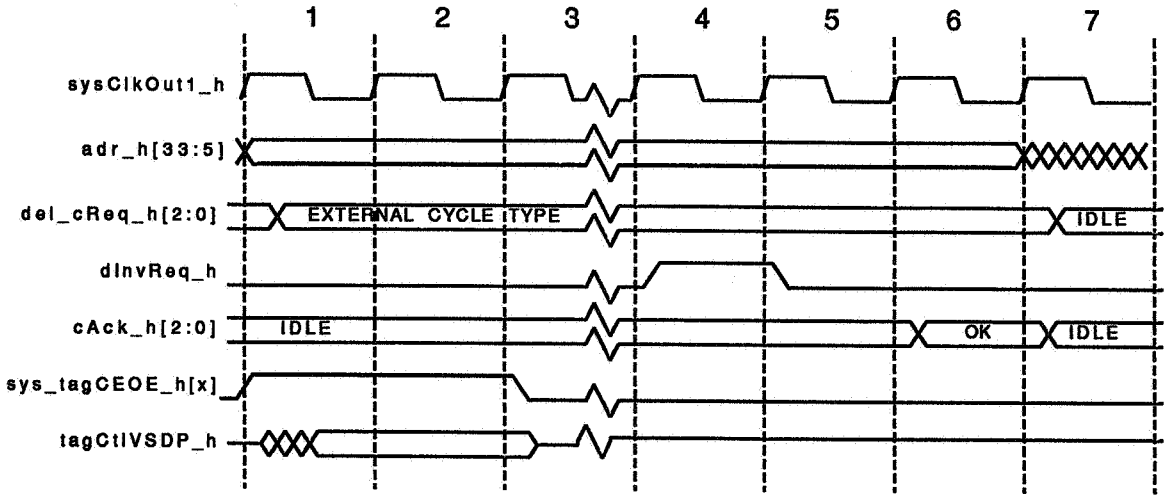


Figure 15 shows the timing for the entire cycle, including the tag control check. The figure shows the delayed **cReq_h[2:0]** lines changing state at the start of cycle 1, and the tag probe occurring during that cycle. The signal **sys_tagCEOE_h[x]** is the system logic version of the 21064 **tagCEOE_h[x]** signal. It is the other input to the NOR gate shown in Figure 3. That nomenclature will be used throughout this application note.

In this case, it was assumed that the Bcache block is invalid, so no victim write needs to be performed. Section 5.5 explains the details of a victim write. If the Bcache probe found that the block was valid but not dirty (that is, it had not been modified since being read from main memory), then the outcome is the same. In both cases, the block can safely be invalidated without a victim write.

Figure 15 shows the tag inspection being implemented in one cycle, so that the read command can start at the beginning of cycle 2. This might not be possible on any particular implementation, and must be carefully analyzed to ensure that the data will be stable when the clock asserts in the system control logic.

The external cycle (READ_BLOCK or WRITE_BLOCK) will overwrite the data in the Bcache, and will assert the **dInvReq_h** signal if appropriate during the fill, so the internal Dcache block will be invalidated later. The lower address bits are directly connected to the **iAdr_h[12:5]** invalidate address input lines in this example, and that will ensure that the correct cache block will be invalidated. Some implementations might want better control over the invalidate bus, and must ensure that the **iAdr_h** lines accurately reflect the lower index value on the asserting edge of **sysClkOut1_h** that samples **dInvReq_h**.

5.3 Read Block Request

If the external cycle is a READ_BLOCK, a 32-byte block of memory information is returned to the 21064. The external logic has complete control of the 21064 pin bus during the transfer. The data is returned to the 21064 and simultaneously loaded into the Bcache. It is the external logic that writes the data into the Bcache during the read cycle, *not* the 21064.

The minimum amount of data that can be written to the Bcache is 32 bytes, but the system logic controls the Bcache until the **cAck_h[2:0]** lines are changed from their IDLE state. As such, it can load and validate more than that if the system designer believes prefetching more blocks is appropriate. Any prefetching must be done in 32-byte increments.

The external logic is responsible for loading the tag address and the tag control fields of the Bcache (with correct parity on both) along with the data. The tag control field should be written as VALID and CLEAN.

Figure 16: Tag Access and Write Circuit

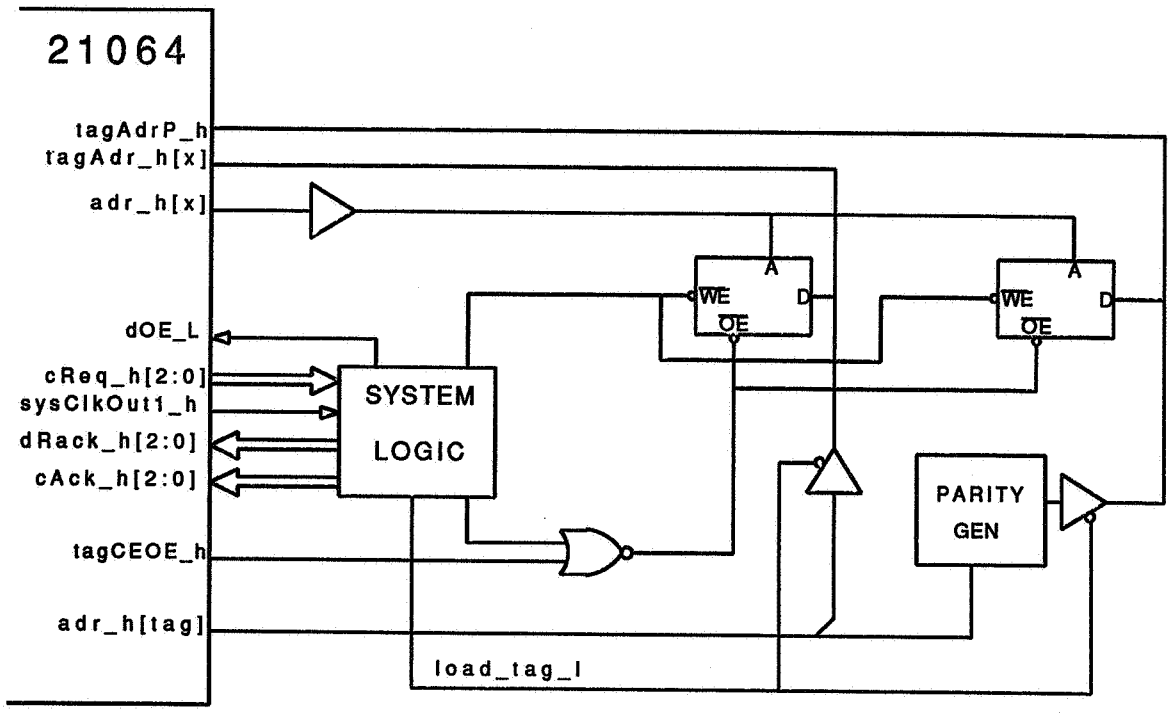
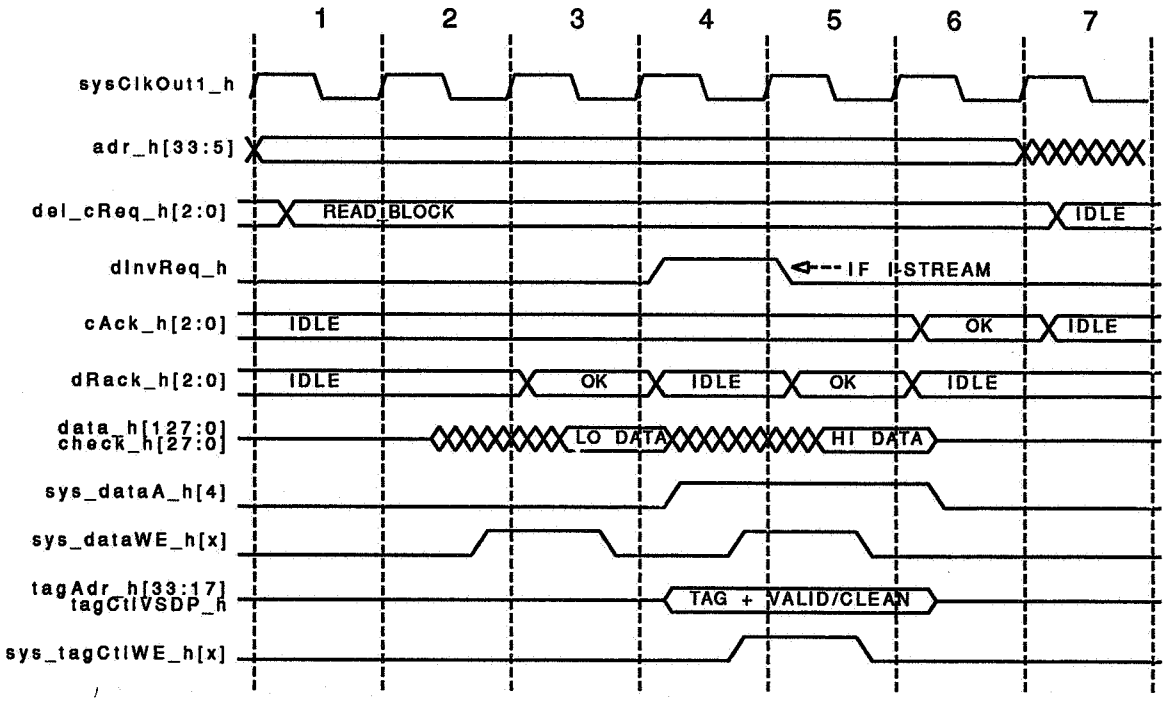


Figure 16 shows an example of the logic that is expected for tag address control. One of the **tagAdr_h** lines is shown connected to its Bcache RAM. All the tag address lines in use by the implementation go to the parity generator. The **tagAdr_h[tag]** signals and the **tagAdrP_h** parity lines are all driven by tristate buffers. On probe reads, the SRAM output enable allows the Bcache to drive the signals, where they are compared by the 21064 or the system logic. On a fill operation, the **load_tag_l** signal causes the Bcache tag RAMs to be loaded with the upper address bits. Notice that the RAM write enable input is not connected to the 21064, since the processor never writes them.

As each data word is returned to the 21064 the **dRack_h[2:0]** field is changed from IDLE to non-IDLE. Normally, the non-IDLE state will be OK, which instructs the 21064 to both check the ECC (or parity) on the returned data and cache the data internally. The preliminary data sheet provides more information on the **dRack_h[2:0]** field. Figure 17 is a timing diagram for a READ_BLOCK data transfer, showing the 21064 control signals.

The data in the example is assumed to be ready at the start of cycles 4 and 6, but in another implementation the data might be ready before or after that time. The **dRack_h[2:0]** lines should change to the non-IDLE state whenever the data is ready, with enough setup time so that they are sensed by the 21064 at the assertion of **sysClkOut1_h**. The **cAck_h[2:0]** lines can also change to their non-IDLE state (signifying the end of the cycle) during the last **dRack_h[2:0]** data phase if desired.

Figure 17: Timing Diagram of READ_BLOCK Cycle



The timing of the Bcache write signal might be tight in relation to the data arriving from the memory. If the memory is a DRAM array, for example, the CAS signal should be de-asserted as quickly as possible after the DRAM data is stable in order to start the next memory access during a page mode read. The Bcache, however, might need the data held stable. Using a bidirectional clocked memory data transceiver, as shown in Figure 2, can help in some cases.

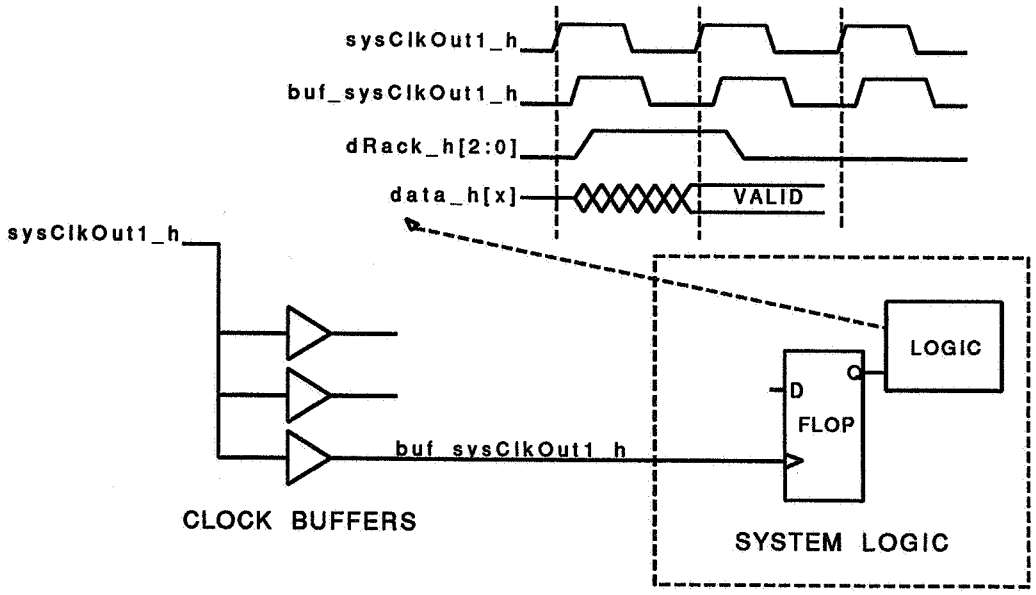
Figure 17 shows the **dInvReq_h** signal asserting, which will invalidate the internal Dcache block corresponding to the lower index bits in the address. This is only needed if the external data fetch is for I-stream (indicated by a false **cWMask_h[2]** on READ_BLOCK cycles), and if the internal Dcache is being kept as a subset of the Bcache. The block that is being filled into the Bcache might be in the Dcache, and it will not otherwise be invalidated on an I-stream fetch.

The 21064 can potentially drive its own Bcache control signals a few CPU cycles into the external cycle. As such, the Bcache SRAMs might still be driving the data bus as the external cycle starts. On a read cycle, the system logic might turn on its own data transceivers early in the access, and should be aware that a system cycle should be allowed before this is done. This eliminates any tristate overlap between the SRAMs and the data transceiver.

For a system without a Bcache, the 21064 signals would be the same as Figure 17, but none of the Bcache related lines would be asserted by the system logic. If the system has a Bcache but it is not enabled, the external system logic needs to have some mechanism to turn off the Bcache fill logic, since the 21064 does not broadcast its internal Bcache enable signal to the external pin bus.

If the read cycle is to an area of memory that has been defined as I/O, it is likely that another bus is involved with the transfer. In this case, the timing is also similar, and the Bcache control signals are also not asserted. A further modification in this case might be to change the **dRack_h[2:0]** field to indicate that no error checking be performed and that the data should not be loaded into the internal chip Dcache either.

Figure 18: Clock Skew From System to 21064

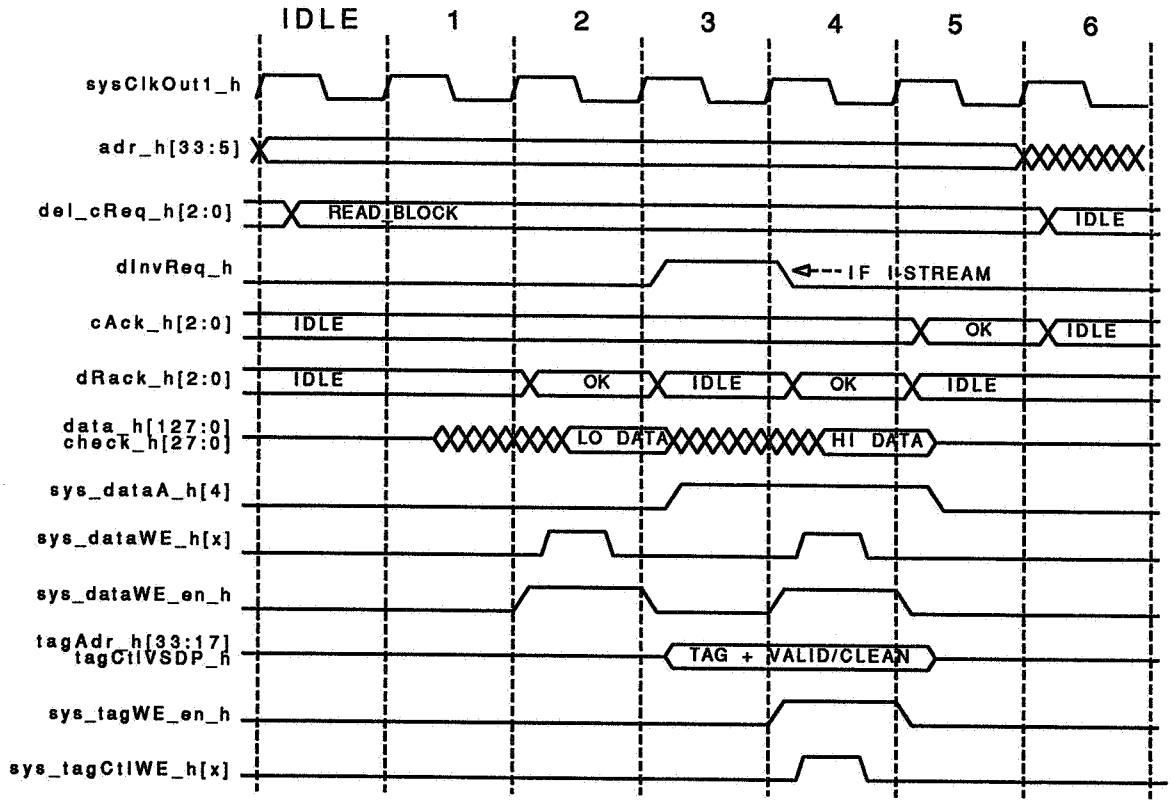


The 21064 system clocks, such as **sysClkOut1_h**, are specified to drive only 40pf. Because of this, clock buffers will normally be used to drive the system logic. The clock buffers add skew between the 21064 and the system logic. Figure 18 shows a timing diagram and a small circuit section that might be used to create the signals in the diagram. The buffered version of the system clock **buf_sysClkOut1_h** drives the system state machines that eventually cause the **data_h** lines to be valid at the 21064 input pins.

The **data_h** must be setup at least 3.5ns before the assertion of **sysClkOut1_h**. In this example, the delay added by the buffer must be added to that setup time, since the 21064 sees its reference clock some time before the system logic. This delay should include the entire path for the buffered clocks, including wire delay, device propagation delay, simultaneous switching increases, transmission line effects, etc. The example in Figure 18 shows only one instance of this consideration. Others must be analyzed based upon the implementation.

It should be noted here that the skew helps signals like **dRack_h[2:0]** and **cAck_h[2:0]**, since they can be asserted on the system logic version of the clock and meet both the setup and hold times in reference to the 21064.

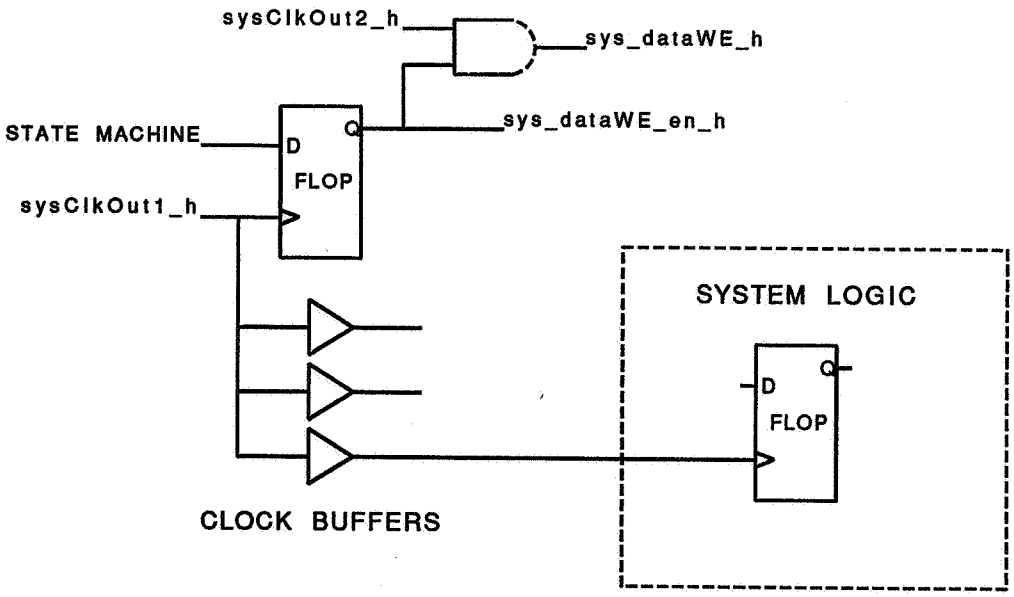
Figure 19: READ_BLOCK Cycle with Write Pulse



If the external timing allows it, a write pulse can be created by delaying **sysClkOut2_h** by 1 CPU cycle, and using **sysClkOut1_h** to create an enable signal for it. Figure 19 shows a READ_BLOCK cycle with a cache fill that uses a write pulse to load the Bcache. The signal **sys_dataWE_en_h** enables **sysClkOut2_h** when the Bcache needs to be written.

Figure 20 is an example of how the write pulse can be created, showing the circuit paths of interest. The clock buffers are shown that are expected to drive the system logic, in part to show that skew must be carefully considered if a write pulse-like scheme is attempted. If the clock buffers add enough delay to the path, and the delayed version of the clock is used to create the **sys_dataWE_en_h** signal, the leading edge of the enable can overlap with **sysClkOut2_h**. To prevent this from happening, a non-buffered version of **sysClkOut1_h** might be used to create **sys_dataWE_en_h**. The same argument applies to the tag control write pulse.

Figure 20: Write Pulse Circuit



5.4 Write Block Request

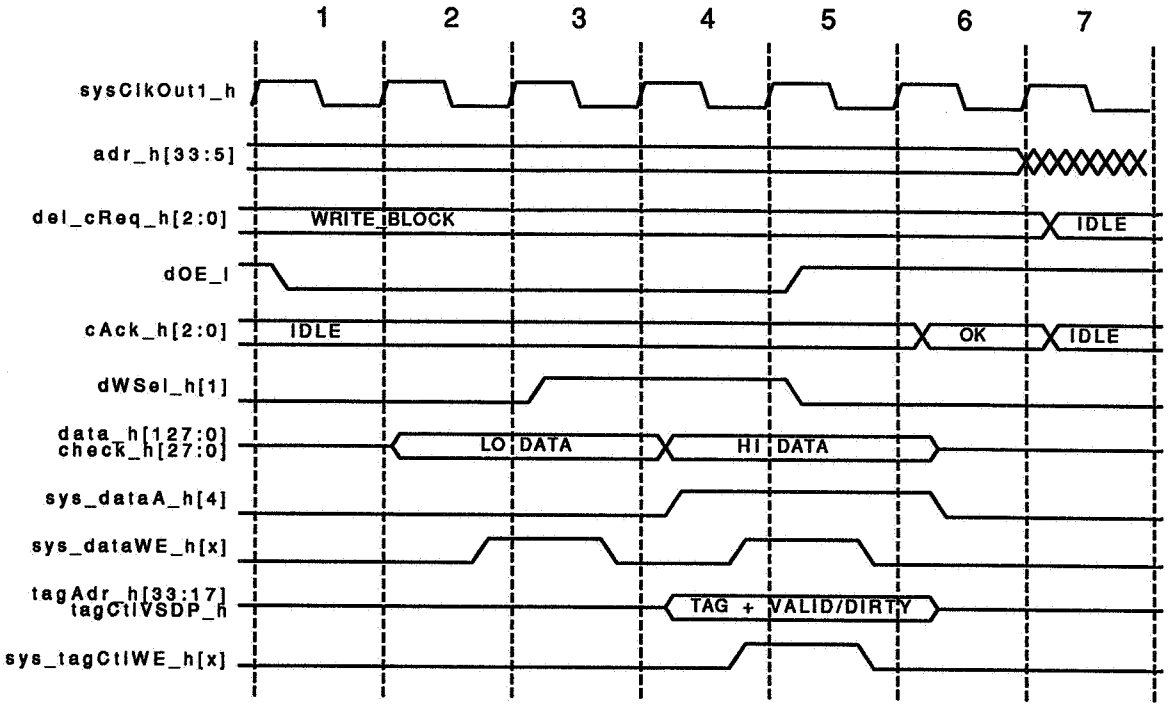
If the external cycle is a WRITE_BLOCK, the system logic must perform a different set of functions. The initial tag probe must still be done by the external logic, and we are still assuming here that the current block is either not valid, or it is valid but not dirty (no victim write needed).

If we assume that the Bcache is used as a writeback cache (the normal mode), and that the design is using a write-allocate Bcache policy, then the write data should go into the Bcache, even though an external WRITE_BLOCK cycle is being executed. The most reasonable way to accomplish this is to read the entire block from memory into the Bcache, then write the masked 8-byte into that same Bcache block. For systems without a Bcache, the external memory should be writable on 4-byte (32-bit) boundaries, since the Bcache merge cannot be performed.

The 21064 is attempting to perform a WRITE_BLOCK cycle in this case, and doesn't even know about the memory read cycle. The **dRack_h[2:0]** and **cAck_h[2:0]** signals should remain IDLE throughout the read transfer. After the read has been accomplished and the main memory data is now in the Bcache block, the system logic should cycle the 21064 through its write data by using the **dWSel_h[1]** line. The 21064 input signal **dOE_l** is used to instruct the chip to drive the data lines for the write portion of the cycle. Only the masked 4-byte segments should have their write enable inputs asserted during the cycle, based upon the **cWMask_h[7:0]** signals.

After the entire read and write cycle have been finished, the tag control should be written as VALID and DIRTY, and the tag address should be written with the correct upper address bits. Figure 21 shows the Bcache write portion of the WRITE_BLOCK cycle. The read

Figure 21: Timing Diagram of WRITE_BLOCK Cycle



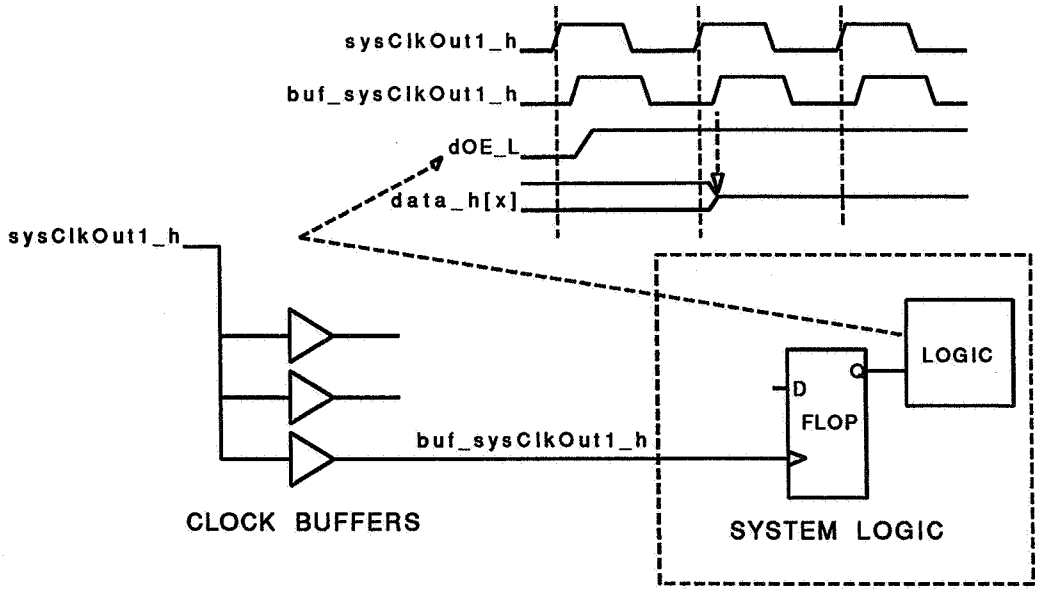
portion looks like Figure 17, except that the CPU acknowledge signals should not be changed from IDLE.

Note when the data actually changes relative to the signals **dWSel_h** and **dOE_l**. All the signals are synchronous to the leading edge of **sysClkOut1_h**, so the inputs are not acted upon until the next system clock edge. The end of the external write in Figure 21 is the start of cycle 7, at which time the 21064 will remove the address and potentially start the next Bcache probe.

There are several optimizations that can be made on the write cycle:

1. The **cWMask_h[7:0]** signals can be inspected, and if they are all set the read portion of the cycle does not have to be performed. In this case, every byte will be written anyway, so the Bcache write cycle can be performed from the start.
2. The tag Bcache RAMs don't have to be written on both the read and write portions of the cycle. It may turn out to be simpler to do it during the read cycle so that it is the same as a normal read, but it is under control of the designer.
3. Both 128-bit data segments don't need to be written if the lower mask bits show that there are no 4-byte segments enabled. The signal **dWSel_h[1]** can be asserted earlier to write the upper 128 bits only. If both segments *are* written, however, the lower address must be written before the upper address (as shown in Figure 21).

Figure 22: Clock Skew From System to 21064 for Write



As with the read cycle, the write cycle must take into account the clock skew between the 21064 and the system logic. Figure 22 shows an example of a potential problem. The 21064 signal **dOE_l** is asserted by the system logic to instruct the chip to drive the **data_h** lines during the write cycle. But **dOE_l** is sampled by the chip on the earlier, unbuffered version of **sysClkOut1_h**. In Figure 22, the data is removed on the asserting edge of **sysClkOut1_h**, which might be too soon. If the system logic uses its version of **buf_sysClkOut1_h** to sample the write data, then it should cause **dOE_l** to remain asserted low one extra cycle to accommodate the clock skew. This same argument applies to **dWSel_h[1]**.

5.5 Victim Write

The second possibility for the original Bcache miss is that the data currently occupying the Bcache block is VALID and DIRTY, but the upper address bits do not match the tag address. The 21064 will go to the external logic with a READ_BLOCK or WRITE_BLOCK, just as in the previous description. When the external logic does the Bcache VALID/DIRTY probe, however, the outcome is different. Since the data in the Bcache block has been modified since it was read from the main memory, it must be written back to memory before the external read or write cycle can continue. The act of writing the block back to memory is called a victim write.

The external control logic for a victim write is straightforward. The 128-bit data segments are read from the Bcache, and the data is sent to the external memory. After the victim is safely back in memory, the READ_BLOCK or WRITE_BLOCK is performed, exactly as described in the previous sections. Some time during the entire cycle (including the victim write and subsequent read or write cycle), the **dInvReq_h** signal should be asserted to invalidate the internal Dcache block for that index.

Figure 23: Timing Diagram of Victim Write Cycle

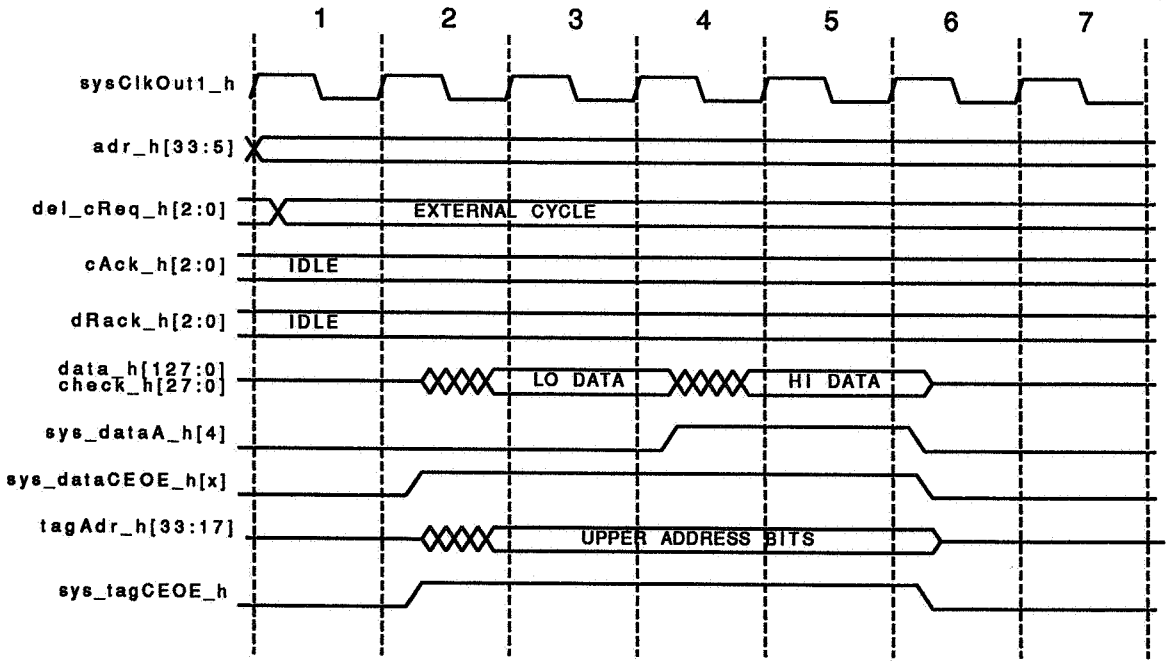


Figure 24: Address MUX for Victim Write

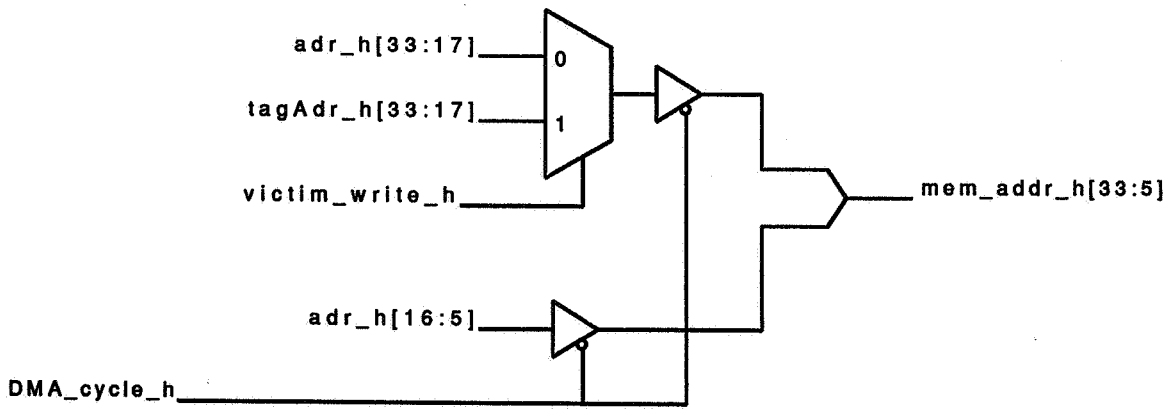


Figure 23 shows the victim write cycle. The tag address is used as the high memory address bits for the write, so the **sys_tagCEOE_h** signal is asserted to enable their outputs. The Bcache data RAMs are enabled, and each data segment is selected in turn by **sys_dataA_h[4]**. In this example, two cycles are necessary for the main memory to be written. If the memory is slower, more cycles should be allocated. At the start of cycle 7 the actual read or write cycle would proceed.

A MUX gate is needed to choose between the normal 21064 memory write and the victim write, where the upper address bits are taken from the tag field of the Bcache. Figure 24 shows the expected circuit. Normally, the MUX selects the **adr_h** lines, but during victim write cycles the **tagAdr_h** lines are chosen as the memory address. Figure 24 also shows that the entire address bus should have the ability to tristate for DMA access. During DMA transfers, the 21064 is forced off the address lines and the external logic controls the entire address. The MUX and tristatable gate can be one physical device.

The signals **victim_write_h** and **DMA_cycle_h** are expected to be created by the system logic. They do not come from the 21064. The tag address field in Figure 24 is shown for the smallest Bcache size. Other Bcache sizes will have different relative widths for the tag and index fields.

For high performance systems, a victim queue (or silo) is an option. Instead of writing the victim and reading the new data word serially, the Bcache and the memory can be read simultaneously. The information in the Bcache can be stored in a silo while the memory data is loaded into the Bcache. The silo can then be used to write the previous Bcache contents to memory.

5.6 Non-cacheable Memory Write

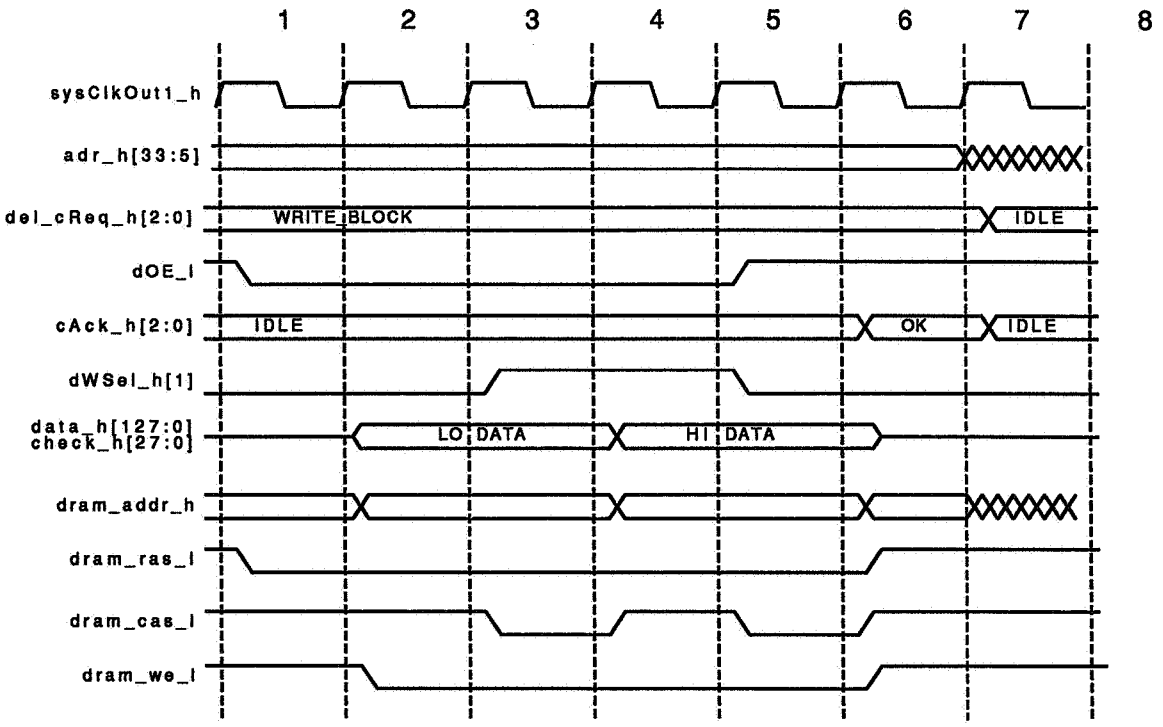
There might be non-cacheable memory space in your 21064 system design. When that area is a write target, the data should bypass the Bcache and be written directly to the system memory. If non-cacheable memory is included in the system, it is best to make it writable on 4-byte segments. Otherwise, a full read/modify/write cycle will be needed to store non-fully masked data.

A memory write on a system that allows masking on 4-byte segments is only a minor variant on the victim write function. The difference is that the information to be written to memory is coming from the 21064 rather than the Bcache. The Bcache is not invoked at all in this situation, and the **dWSEl_h[1]** signal is used to instruct the 21064 about which 128-bit data segment to provide.

Figure 25 shows the timing for such a write cycle. In this example, a more complete memory control flow is shown. It is assumed that the memory is a DRAM array, and a representative set of memory control signals are provided. The designer should work out the exact timing on a particular implementation in order to ensure that the memory parts are accessed within specification.

The **adr_h** lines should be stable at the start of the cycle, since they are changed by the 21064 before the cycle is started. If the DRAM address MUX points to the row address by default, the memory can be RASed at the start of the cycle. At the end of the cycle, the DRAM RAS precharge time must be accounted for. The 21064 will allow at least one idle cycle after it senses **cAck_h** as non-IDLE before it will start the next external command. In the example, RAS de-asserts at the start of cycle 6, which means that it cannot re-assert until the start of cycle 8. The changing of **cAck_h** so that it is sensed at the start of cycle 7 meets the RAS precharge time for the part in this implementation.

Figure 25: Timing Diagram of Direct Memory Write Cycle



6 Load Locked and Store Conditional

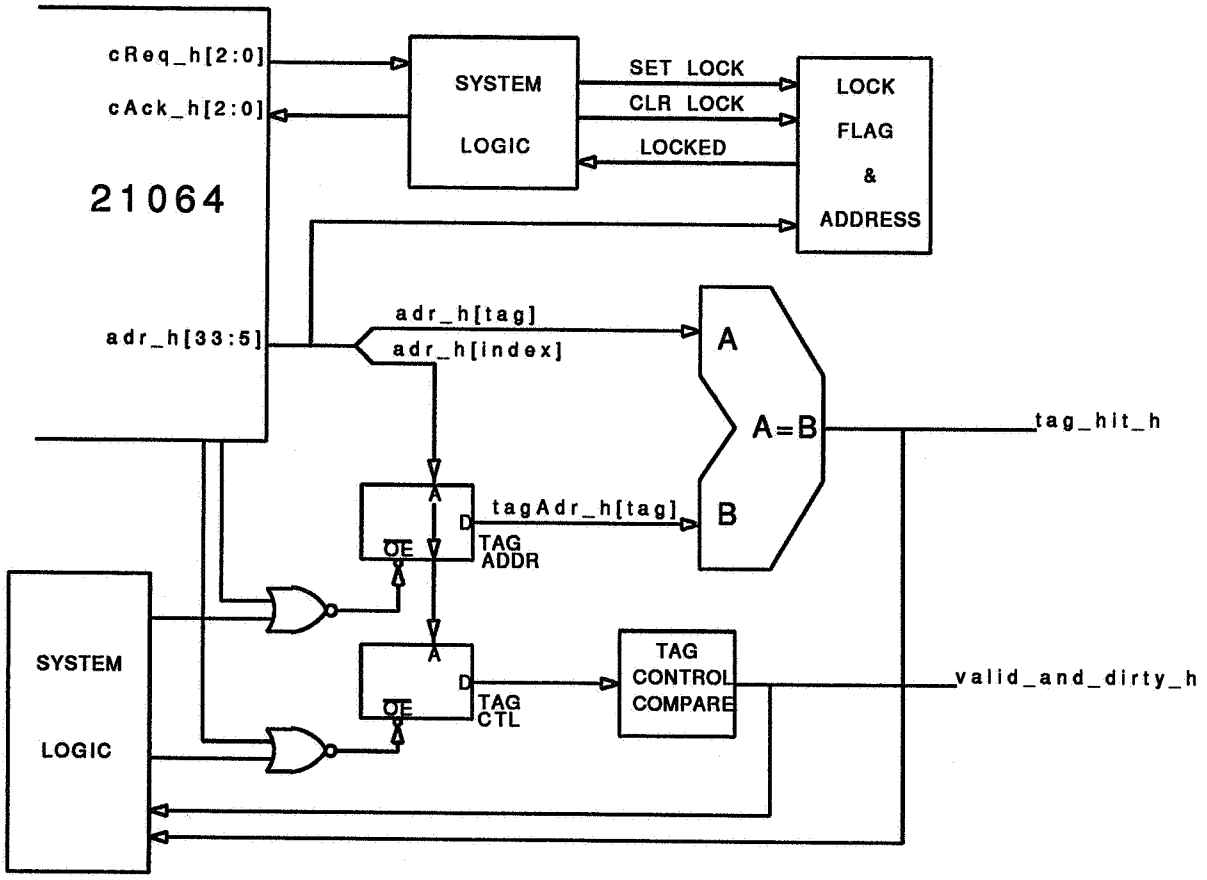
The 21064 provides the ability to perform locked memory accesses through the LDxL (Load_Locked) and STxC (Store_Conditional) cycle command pair. The LDxL command will force the 21064 to bypass the Bcache and request data directly from the external memory interface. The memory interface logic must set a special interlock flag as it returns the data, and may optionally keep other information about the transaction (such as the locked address).

The data requested for the LDxL access might be in the Bcache, since it has not been probed, so the external memory logic must do its own probe to determine where to obtain the information. In previous descriptions, the system logic only had to probe the tag control VALID and DIRTY RAMs to determine if a victim write was necessary. For the LDxL and STxC probe, the entire tag address must be compared, since the data that is being accessed might be in the Bcache.

Figure 26 shows a diagram of the probe and compare logic. On the initial request (the **cReq_h[2:0]** lines specify that the external LDxL must be performed) the system logic enables the tag RAMs and compares them to the tag field of the address for the 21064. If they compare and the block is valid, then the data requested is already in the Bcache. If the tag compare also shows that the block is dirty, then the *only* place the data resides is in the Bcache. There are two choices:

1. The data can be accessed from the Bcache.
2. The data can be written back to memory, then accessed from there.

Figure 26: Tag Address Compare Circuit



If the tag compares and the block is valid, but it is not dirty, then both the Bcache and the memory contain the data. It can be accessed from either place. If the tag fails or the block is *not* valid, then the data is only available from memory and must be accessed from there. In all the above cases, a flag must be set that signifies the location is locked.

Every design needs to provide a lock flag, but the amount of address information latched is completely up to the designer. On a uniprocessor system that does not expect much lock contention, simply having the lock flag with no address information might be enough. If any device accesses a memory location, the flag can be cleared, which will cause the subsequent store cycle to fail. On a multiprocessor system that expects real lock contention, lock address information can be saved so that different processors can lock different areas.

The STxC instruction is executed by the 21064 to clear the lock (and to find out if the code that was executed did so without contention). It is a write-type request where the processor bypasses the Bcache without a probe. If no other access has been made to the locked data, the STxC is treated similarly to a regular external memory write, though the Bcache must be probed by the system logic to determine where the most up-to-date data is located. The locked flag is also cleared.

If the Bcache probe finds that the data is both valid and dirty, the choices are similar to the read case:

1. The data can be written into the Bcache, using the **cWMask_h[7:0]** to determine which 4-byte segments should be modified. The STxC command will never validate more than a single 4-byte or 8-byte segment of data, and this can be used to optimize the cycle if desired.
2. The data can be written back to memory with a victim write, and modified there.

If the locked data location has been accessed between the LDxL and STxC commands, the external memory logic must return a special acknowledgment code that notifies the 21064 of this fact. In this case, no Bcache probe or actual external cycle needs to be performed.

7 Special Request Cycles

There are some external request cycles that might not actually perform any work, but must still provide the 21064 with an acknowledgment. The BARRIER, FETCH, and FETCH_M cycles are described in the preliminary data sheet, and will perform a system-specific function. When they are sensed by the external control logic, the system must minimally provide a **cAck_h** acknowledgment of OK.

8 DMA Access

There are situations where a device connected to an I/O bus needs direct access to the 21064 cache/memory subsystem. In the most general case the data could be in the Bcache, and that is the one discussed in this section. If a restriction can be made that eliminates the possibility of the target data being in the Bcache (that is, the DMA is always done to uncached space), then a simpler version of this discussion applies.

There are several ways that the external logic can perform a DMA access, the most straightforward of which is the use of the **holdReq_h** line. When a DMA device requires access to the 21064 cache/memory subsystem, it can notify the chip of that fact by asserting the **holdReq_h** signal. The 21064 replies to this request by asserting the **holdAck_h** signal. This signifies that the 21064 is no longer asserting the address, data, or Bcache control signals. The entire memory subsystem and Bcache are now under control of the external system logic.

The signal **holdAck_h** changes simultaneously with **sysClkOut1_h**. As such it should be sampled on an edge other than **sysClkOut1_h** if used as an input into state machines that run on **sysClkOut1_h**. This is similar to how **cReq_h[2:0]** must be used, as shown in Figure 14.

If it is assumed that the DMA target data (read or write) might be in the Bcache, the external logic must do a Bcache probe. This is similar to the probe necessary to determine if the data is in the Bcache when a LDxL or STxC is executed. The tag address and control RAMs should be compared to find out if the requested data is in the Bcache, and if it is dirty. The DMA logic can use the LDxL/STxC compare logic shown in Figure 26, or it can duplicate that logic for its own comparison.

The 21064 provides a third option for the tag address comparison, and this is the **tagEq_l** signal. When the chip is in **holdReq_h** mode, the **adr_h[33:5]** signals become inputs. The DMA device can drive its address on those lines and simultaneously enable the tag address RAMs. If the tag address compares with good parity, the signal **tagEq_l** will be asserted low. Consult the preliminary data sheet for more details about the use and timing of this feature.

For DMA read cycles where the probe shows that the data is valid in the Bcache, the choices are similar to what they were for the LDxL/STxC probe. If the data is valid but not dirty, it can be accessed from wherever it is most convenient. If the data is valid and dirty, it can be accessed directly from the Bcache or written back to memory and accessed there.

For a DMA write that hits in the Bcache, there are several choices:

1. The data can be written directly into the Bcache with the correct ECC or parity. In this case, the tag control should be made DIRTY, and the **dInvReq_h** signal should invalidate the cache line in the internal Dcache.
2. The data can be written back to memory with a victim write, and it can be modified there. The **dInvReq_h** signal should be asserted during the victim write or the DMA memory write to invalidate any stale Dcache data.

If the Bcache probe misses, or if the DMA access is defined to be only in the memory, then it is most sensibly accessed or modified there.

After the read or write cycle is complete, the **holdReq_h** signal can be de-asserted, which will cause the 21064 to de-assert the **holdAck_h** signal. The 21064 will then take control of the bus again, after a short delay.

There is one subtlety that should be mentioned here in regard to DMA access design. The 21064 might be in the middle of its own external (non-Bcache) access when it receives the **holdReq_h** request signal. If this happens, the chip might be waiting for data of its own, and has really only stalled the external cycle. As such, the data and cycle acknowledge signals are "live". The external logic must be careful not to assert the **dOE_l**, **dWSEL_h**, **dRack_h**, or **cAck_h** signals during its access cycle. Furthermore, there is a 2-CPU cycle delay between the time that the 21064 de-asserts the **holdAck_h** signal and when it re-enables its own address and data lines. This must be factored into the external logic for cycles that continue after the DMA stall.

In order to simplify the design, it is possible to filter the **holdReq_h** signal going to the 21064. If the external logic ensures that the **holdReq_h** signal only gets to the 21064 between cycles, then the problem of external cycles stalling in the middle is eliminated.

9 Backmapping the Internal 21064 Dcache

The 21064 provides the ability to keep a "backmap" of the internal Dcache tag address in external logic. In effect, the module adds enough extra information about the Dcache tag address to filter the invalidates that are sent to the 21064 Dcache. This can be used in multiprocessor systems or to filter DMA writes.

The processor outputs the signal **dMapWE_h** when it loads a block into the Dcache. This is meant to control an external memory array that takes the address from the appropriate **adr_h** lines and updates the external tag address memory location.

The external tag address does not have to contain the entire Dcache tag field, but rather needs only the difference between the Bcache and Dcache tag widths. If the Dcache is being kept as a subset of the Bcache, and if the Bcache is first probed, then the Dcache backmap is only responsible for knowing if a Bcache hit is also a Dcache hit. The preliminary data sheet describes the backmap in more detail.

10 I/O Interface

The input/output function of the 21064 is in some ways a subset of the memory function. I/O is normally not cached, so the probe will miss or not be performed at all for that memory quadrant. The access will go directly to the external interface bus as a READ_BLOCK or WRITE_BLOCK.

On a read cycle, the data will be returned as in the memory access already described, with the **dRack_h[2:0]** signals indicating that the data should be neither error-checked nor cached inside the chip. Since the return data is under complete control of the system interface logic, the Bcache will not be filled. On a write cycle, the steps are similar to a direct memory write cycle. The external logic can take the appropriate number of data words, then acknowledge the cycle.

The Alpha architecture provides an approach to I/O called a "mailbox". A description of the read or write is set up in memory. The description includes the full address, data, and mask information. A special mailbox register is then accessed in order to invoke the I/O transaction. This approach implies a smart I/O controller, and allows access to the full address range of the I/O bus.

If the mailbox option is not implemented, there are some techniques that can be employed when interfacing the 21064 to an I/O bus:

1. Address or data bits can be used to create byte masks and encode system level functions.
2. The 21064 address lines **adr_h** can be shifted right when accessing external buses that need the lower address bits. So, for example, **adr_h[20:5]** can translate to I/O address bits [15:0].
3. Reads and writes to I/O space can use the low bytes for all transactions, rather than pack the data into the appropriate field within the 32-byte block.
4. The **cWMask_h** field can normally be ignored for I/O writes.

11 Summary

The intent of this application note was to provide information so that a logic designer could understand the fundamental principles of creating a system with the 21064 processor chip. All of the chip's features were not covered, and it is suggested that the preliminary data sheet be consulted for more details about the 21064 and its use. Future application notes will cover different aspects of 21064 system design.

JJ

"ALPHA" technology on test at the University of Edinburgh

Brian Murdoch

(Extract from Edinburgh IT Forum, Autumn 1992)

Editor: Frances Allen,
Computing Services,
Main Library,
George Square,
Edinburgh
EH8 9LJ

tel: 031 650 3322

email: Frances.Allen@ed.ac.uk

"ALPHA" technology on test at the University of Edinburgh

Brian Murdoch

Since August, a group of colleagues from around the University has been participating in a field test for the Digital Equipment Corporation on their new ALPHA system. The performance of this system has elicited comments like "amazing" and "brilliant" from researchers who already use very competent computing facilities. The system "hit the floor running": we were soak-testing it with ported code within two days of its being handed over.

High technology in Scotland

For some time Digital has been working on a new design of microprocessor architecture (essentially, instruction set), and the necessary chip fabrication processes to bring their new design into production. This design, code-named ALPHA, was announced on 25th February this year, simultaneously in the USA and at South Queensferry - where Digital has invested £250m in a plant to fabricate their latest chip designs, including ALPHA.

This ALPHA chip has been built into systems which were announced on 10th November, ranging from desk top workstations to 'data centre' sized systems. As with their previous VAX design, all the ALPHA systems use the same architecture and operating system, maintaining complete compatibility throughout the range. The machines run Digital's own VMS system, the OSF/1 variant of Unix, or Microsoft's Windows New Technology (Windows NT). Some of these machines have been built into existing hardware infrastructures, presumably reducing costs and bringing forward their availability. It also means that certain recent VAXs can be upgraded to ALPHA systems simply by replacing the processor board.

Unlike Digital's VAX design, the ALPHA chip is openly on sale at any level of integration, from the bare chip itself, through processor boards to complete systems. Digital will also license VMS to third parties so that they can build ALPHA based VMS systems to sell independently. Some other system suppliers, such as Cray, have already signed up to take advantage of these opportunities, their systems probably running OSF/1.

The ALPHA chip is certainly a very significant investment and achievement for Digital. The chip is clocked at up to 200 MHz depending on model of system, and is certainly the fastest commercially available chip at the time of writing. Some of its principal features are:

- 64 bit RISC multiple-instruction-issue
- 0.75 micron CMOS
- Clock frequency up to 200 MHz
- Device count: 1.7 million
- 291 Signal pins
- On-chip caches: 8 Kbyte each for data and instructions
- Floating point unit: on-chip IEEE and VAX formats
- Pipelines: 7-stage integer, 10-stage floating point

These ALPHA chips are now being made at the South Queensferry fabrication plant. It is perhaps worth noting that Digital has also moved its 50-strong UK VMS software engineering team to Livingston. Add to that its Ayr assembly plant, and one can see that Digital has invested hugely in development and manufacture in Scotland. Interestingly, while passing through the VMS engineering offices the other day, I was surprised to see many faces of students I remember roaming the Computer Science corridors at The King's Buildings. Even more interesting was meeting Bill Laing, who used to work in both that department and the EUCS (ERCC as was), and who contributed so much to EMAS. From his Scottish base in Livingston, Bill is now the (worldwide) VMS Technical Director, spending much of his time in America or working American hours when in Scotland.

Field testing at EUCS?

In February this year, Digital invited the University's Computing Services to submit a proposal to be a field test site for an ALPHA system. After consultation with the EUCS Directorate and other colleagues, it was decided that we could carry out such a test thoroughly, if it ran the VMS operating system. Running the test on the OSF/1 Unix variant was also a possibility, but the Unix support team were fully, or even over, committed with new developments, without the time and effort available to test fully a new variant of Unix.

On the other hand, VMS is VMS is VMS, and we in the VMS support team were confident that we could organise activities on the systems we manage so as to make room for a field test: we were confident that ALPHA/VMS would be sufficiently close to VAX/VMS in matters that concern us that there would be little effort in running up and managing the system. We would thus be able to devote our efforts to managing the field test. The core of the test would be to port substantial software applications to the ALPHA/VMS system.

Once it had been decided in principle to submit a proposal, we had two weeks to find suitable software porting projects (from the 'heavy' computing interests around the University), assemble other background information, and compose a cogent and informative proposal. This just would not have been possible without "using the technology". Requests for projects were made via electronic mail - and a skeleton proposal form sent with the request. The responses were merged using a text processor, producing half of the total proposal in less than a day. Finding the required background information was much more difficult, and involved phoning many people in the University to gather all that I wanted.

However, we did submit all this to Digital, with two days to spare, and it caught the eye of the ALPHA field test managers in the USA. After months of waiting we heard in June that we had been selected as the only site in Europe to be assigned the heaviest system in the ALPHA range, on which to test ALPHA/VMS. The only other ALPHA/VMS test site in the UK is Rutherford Appleton Labs. World wide there are about 30 test sites.

We received a visit on the 9th of June from one Vaughn MacKie, head of the ALPHA field test programme. He told us that the proposal that this University had assembled was the most comprehensive and informative that they had received, from anywhere, and for this I must thank my colleagues throughout the University who responded so promptly and cooperatively to my electronic form filling request. There is no doubt that their timely efforts were crucial to this University being seen as a well organised site in which Digital could have confidence.

The new system arrives...

The system was delivered on 10th August, and consists broadly of:

- ALPHA processor - 180 Mhz (Max of four processors)
- 256 Mbytes main memory (Max of 14 Gbytes)
- 12 Gbytes disk space
- CD ROM, DAT tape, ethernet connection etc.

The purpose of the field test is to assess features such as:

- Ease of porting applications to ALPHA/VMS
- Correctness of the converted code
- Reliability of the system
- Performance of the whole system, not just the processor
- Digital's ALPHA support procedures

...and goes to work

The applications that colleagues offered as porting exercises forming the basis of the field test were as follows:

Name	Department	Application Area	Package
Paul Taylor	Biochemistry	Molecular dynamics	X-Plor
David Taylor	EUCS	Geophysical modelling	ANISEIS
David Richards	Physics	Lattice Gauge Theory	Spectrum
Wyn Williams	Geo. Sciences	Magnetic recoding	Micromag
Douglas Heggie	Maths & Stats	Stellar dynamics	NBODYI
Rob Pooley	Computer Science	Simulation modelling	SIMULA
Ken Peach	Physics	Monte Carlo Simulation	NMC
Michael Palmer	Chemistry	Molecular structure	GAMESS-UK
Martin Connell	Medical Physics	3D image processing	
Brian Main	Economics	Economic modelling	LIMDEP
Nick Hulton	Geography	Ice sheet modelling	EISM
Dean Livelybrooks	Geo. Sciences	Electromagnetic induction 3Dfeem	

("Geo. Sciences" is the department of Geology and Geophysics)

The main effort of the test has been shouldered by the 12 individuals named above, and their colleagues, who have done the actual porting of code on to the ALPHA system. Jason Crain of Physics also started porting a computational condensed matter application about a month ago. The EUCS has had a minor role in the actual porting work, except for John Blair-Fish and Naresh Mooljee who have been assisting a couple of the projects. The EUCS's main role, being covered by the VMS Support Team, has been to organise and manage the field test and to provide support to all these projects.

Digital has expended considerable efforts to make the porting of software to ALPHA as easy as possible. It is inevitable that moving VMS itself to a new

architecture would involve some changes. However, it is remarkable that we have met almost no difficulties associated with the system itself. This is reflected in the fact that many third-party software suppliers have ported their applications to ALPHA/VMS, and have even stated on the Internet just how straightforward the process has been (for example, TGV Software - who sell a TCP/IP package for VMS). Of our own projects, the first one (consisting of several thousands of lines of Fortran) simply compiled and ran. The only difference was the astonishing speed with which it executed. A couple of projects with greater external dependencies have taken some time to make progress, but others have been relatively painless, especially compared with some other systems with which those doing the porting have been involved.

One especially neat mechanism that Digital has developed is a utility called VEST, that takes a VAX/VMS executable module and converts it into an ALPHA/VMS executable module. Barring code that uses particularly esoteric system facilities or relies on aspects like the memory addresses of system data structures, the technique works. One would want to use it, probably as a temporary expedient, where a suitable compiler was not already available on ALPHA, where the source code was not available, or where the re-compilation for ALPHA had run into problems. This technique does work and we have used it ourselves, especially to get some of the non-VMS editors running quickly on ALPHA/VMS. It was also used to convert a utility for reading Unix TAR tapes.

A Good Result

We are now nearing the end of the 90 day field test. Over half of the projects have completed their porting and are using the system for real science. Those involved in the completed projects can now see how fast this system can perform, and are delighted to have a system of such power at their disposal, for the time being. It looks as though more than one scientific paper will be produced in the near future as a result of the researchers having this ALPHA system available to them. At least two of the projects are using the system as a resource complementary to the parallel systems in the University.

When we were first discussing this field test with Digital, we were warned about two things. Firstly, only 50% of field tests turn out to be satisfactory from Digital's point of view. However, we have already received a note from the field test managers in the USA complimenting us warmly on what we have achieved to date - the implication being that we have established a basis for further cooperation with Digital. Secondly, participating seriously in a field test is hard work - and on behalf of all my colleagues, I can confirm that. Of course, any thoughts of further work for Digital in this field must be considered in a business-like manner: the University must now take a view of the balance of effort against benefit. There have been two attempts to establish a broader relationship between Digital and this University in recent years, but neither worked out. This latest field test initiative, not involving any specific 'up front' commitment on either side, yet providing ample scope for establishing a practical relationship and generating goodwill, could provide the stimulus that previous mutual approaches failed to find.

I must thank all of the project collaborators and their colleagues who have worked so hard in making a success of this field test. My thanks also go to my colleagues in the VMS Support Team - Joss Bartlett, Angela Lamb and Edna Walton: they have been heavily involved in this field test while ensuring that the VAX/VMS services kept running. Nor must I forget the support I received for our participation in this field test from senior colleagues in the EUCS - Richard Field, Brian Gilmore and Andrew McKendrick. And lastly, our thanks are due to all in Digital who have supported our efforts over the last three months.

Computing Services

The University of Edinburgh

at

James Clerk Maxwell Building
The King's Buildings
Mayfield Road
Edinburgh
EH9 3JZ

and

Main Library Building
George Square
Edinburgh
EH8 9LJ

DECchip™ 21064 Microprocessor

Heat Sink Assembly Instructions

digital™

Heat Sink Assembly Process Description

Tools Required:

Vacuum pick-up tool
Manual nut/torque driver
Available from Contact East
Contact - Paul Shae (508) 682-2000
Parts: TS30 torque driver
1/4" hex bit
3/8" socket with 1/4" hex drive
Nut placement tool (optional)
Available from Advanced Torque Systems
Contact - Warner Neibel (603) 472-2431
Parts: T-1E10 1/4" drive 5/16" hex socket
EX-250 1/4" drive hex 3" long

It is recommended that the heat sink be installed after the package has been assembled to the PWB. The process steps are as follows:

1. Attach a part label to the heatsink if desired. The user may want to transfer date codes or record other markings from the package so it can be viewed on the fully assembled module.
2. Place module on a holder.
3. Place Grafoil onto the package slug (using vacuum pick-up tool).
4. Place Heatsink on top of package slug.
5. Place aluminum nuts on the threaded posts using a nut placement tool.
6. Tighten nuts to the specified torque level of 15 inch-pounds (+/- 2 in-lbs) using a manual nut/torque driver.

Material Description

Grafoil - Flexible Grafite Foil Preform

- The Grafoil sheet is used to improve the thermal conductivity between the package and heat sink by replacing micro-air pockets with a less insulative material.
- During the heat sink assembly process, a Grafoil sheet is placed on the slug of the ceramic package.

Heatsink

- The heat sink material is 6061 aluminum T-6 and the heatsink finish requirement is clear anodization, hot water sealed, per MIL-A-8625C, Type II, Class I.

Nuts

- The nut material is 2011 T3 aluminum (this grade is critical). The nut style is hex machine screw, double-chambered, double-countersunk per ANSI B18.6.3. The thread size is 10-32, Class 2B.

Detailed Assembly Process Steps

Heat Sink Labeling:

- No specific labeling is recommended. Labeling before assembly of the heat sink to the package is recommended. The user may want to transfer date codes or record other markings from the package so it can be viewed on the fully assembled module.

Module Set-up:

- It is recommended that a fixture be used to hold the module in place during heat sink assembly to the package.

Grafoil Pick and Place:

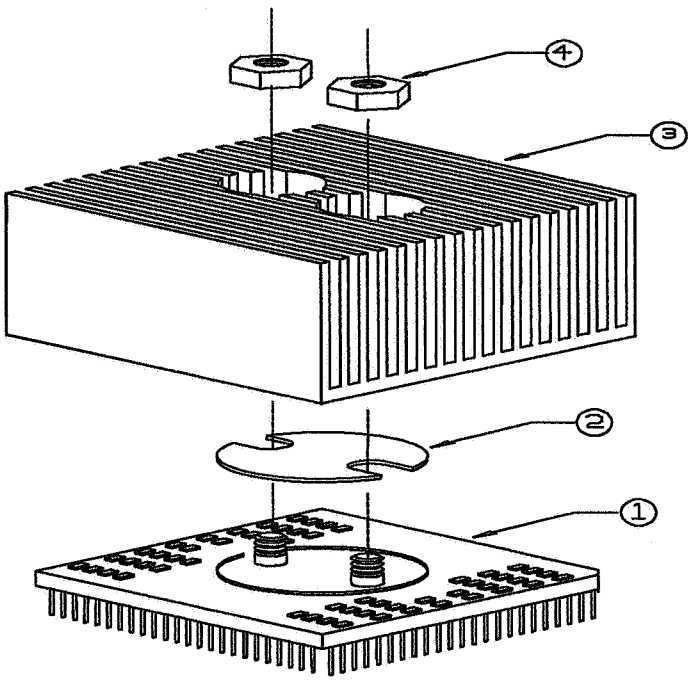
- Perform a visual inspection of the package slug to ensure it is free of contamination.
- Pick up the Grafoil at the center using a vacuum pick-up tool. **(Do not do this by hand.)**
- Place the Grafoil on the gold plated slug surface and align with the threaded studs.

Heat Sink Attachment:

- Align the heat sink holes to threaded studs on the ceramic package.
- Place the heat sink by lowering it onto the component (handle by the edges).
- Insert a nut into the placement tool and place on stud until the nut separates from the tool.
- Set calibrated torque driver to 15 in-lbs (+/- 2 in-lbs) (metric: 1.7 Newton-meters, +/- 0.2 N-m) and tighten.

Heat Sink Removal/Replacement

- If removal/replacement of the heat sink becomes necessary, remove Grafoil completely, clean slug and heat sink surface with Isopropyl alcohol, and replace with new Grafoil and nuts.



KEY:
1 = DECchip 21064
2 = Grafoil
3 = Heat Sink
4 = Aluminum Nuts

Heatsink Performance

Heatsink dimensions: 2.69" x 2.69" x 1.35" h. (68.3 mm x 68.3mm x 34.2 mm h.)

Below is provided a reference of heat sink performance and maximum allowable ambient temperature. If you need to calculate maximum ambient temperature at other frequencies, please use the following formula:

$$T_a = 85 - [23 / (150 * f * \theta ja)]$$

where f is measured in MHz.

Air Velocity	THETA Junction-Ambient θja (max)	Maximum T_a (measured at package) @ 23W
50 lfpm	3.25 Deg C/Watt	10.0°C
100 lfpm	2.50 Deg C/Watt	27.5°C
200 lfpm	2.30 Deg C/Watt	32.0°C
300 lfpm	2.00 Deg C/Watt	39.0°C
400 lfpm	1.90 Deg C/Watt	41.0 °C
500 lfpm	1.80 Deg C/Watt	43.5 °C
600 lfpm	1.75 Deg C/Watt	44.5 °C
700 lfpm	1.70 Deg C/Watt	46.0 °C
800 lfpm	1.68 Deg C/Watt	46.5 °C
1000 lfpm	1.66 Deg C/Watt	47.0 °C

For more information about Alpha and the heat sinks contact our Hotline:
1-800-DEC-2717
1-800-DEC-2515 (Telecommunications Device for the Deaf --TDD)
1-508-568-6868 (local number)
1-508-568-6447 (FAX)

For more information on how to order Alpha, contact one of Digital's Sales Representatives. Call your local Digital sales office for details.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Copyright (C) 1992 by Digital Equipment Corporation. All Rights Reserved.

Printed in U.S.A.

The following are trademarks of Digital Equipment Corporation:
DIGITAL, DECchip, and the DIGITAL logo.

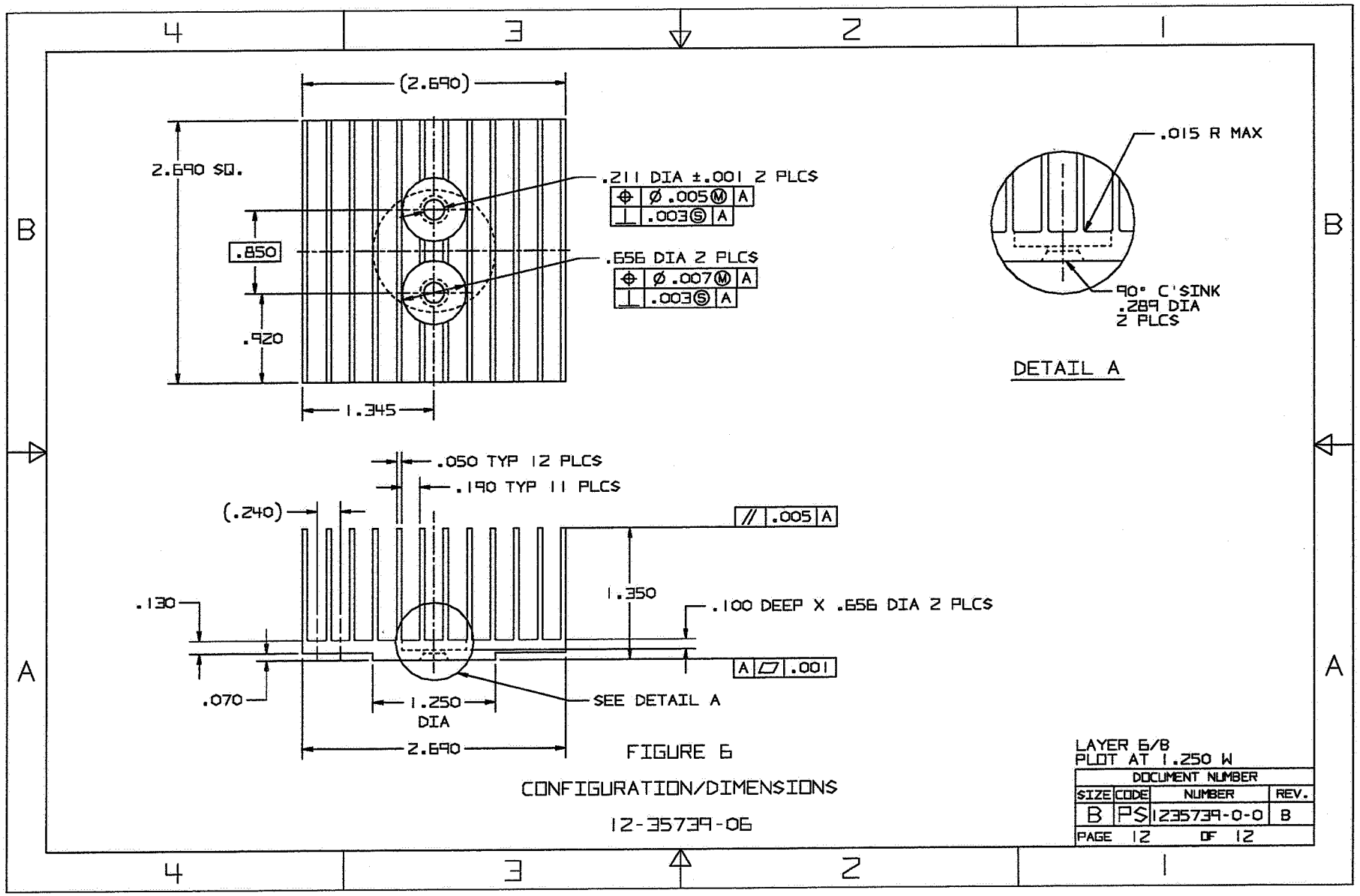


FIGURE 6

CONFIGURATION/DIMENSIONS

12-35739-06

LAYER 6/8
PLOT AT 1.250 W

DOCUMENT NUMBER			
SIZE	CODE	NUMBER	REV.
B	PS	1235739-0-0	B
PAGE 12		OF 12	

Edniburgh University Computing Services

selected for

Extended ALPHA Field Test

In June last year, Edinburgh University Computing Services (EUCS) was selected by Digital as the only site in Europe to field test one of the new DEC 7000 ALPHA systems running the VMS operating system. The tests started in August and ran for three months, during which time substantial scientific applications were ported to the system, many running with a speed that astonished the researchers involved.

The results of these tests proved to be so satisfactory to Digital that they have invited EUCS to participate in an extended field test programme for the next eighteen months. This type of offer has not been made to any other site. This is a considerable achievement for the University I must express my thanks to all of those who participated in the initial field test.

As new DEC 7000 hardware components become available from Digital Engineering, they will be incorporated into the system and exercised by the varied work load that is already building up and the reliability and performance reported back to Digital. It is not possible to reveal details of the products that will be under test as most of them are not yet announced. However, it is expected that the programme will start with a 90-day multi-processor test, probably involving four 180 MHz processors, more memory and more disks (for extended page files).

While our colleagues, especially those who participated in the initial field test, reap the benefit of having this system available to them, the EUCS has now to try and recover some of its investment in time and effort. We hope to achieve this through encouraging commercial and industrial interests to use the system and our accumulated ALPHA migration experience for familiarisation and migration work. This style of activity is an opportunity to forge new links between industry and commerce and the EUCS which could develop to provide wider and longer term mutual benefit.

Brian Murdoch, EUCS, 20th January 1993

AXP HAMILTON RENTALS

Alpha Link

bringing you access to DEC's new technology

Hamilton Rentals, in conjunction with EUCS Enterprise, brings you bookable access to Digital's *Alpha*/VMS system through the *Alpha* Link service. This will allow you to gain real experience of the system to try your own software or to port existing software.

EUCS Enterprise participated in the original field test programme, running on a DEC 7000 model 610 (data centre server), being the only site in Europe selected for this class of machine. They have now, uniquely in the world, been made the subject of an extensive rolling field test programme. This test programme will include multi-processor testing and more new hardware products as they emerge, extending their already large knowledge of migration to *Alpha*.

Hamilton Rentals believe that EUCS Enterprise have unrivalled early experience and, through the rolling field test programme, will remain at the forefront of this new technology.

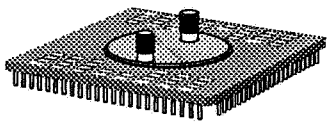
To find out more about how to use *Alpha* Link, phone Tom Frood on 0506 416911, or fax to 0506 416933.

Digital 21064-AA Microprocessor

Product Brief

Preliminary, February 1992

digital™



Features

- Full 64-bit Alpha architecture:
  - Advanced RISC architecture
  - Optimized for high performance implementations
  - Multiprocessor support
  - IEEE single and double precision, VAX F_ floating and G_floating, longword and quadword data types
  - Cycle counter for code optimization
- Privileged Architecture Library Code (PALcode) supports:
  - Optimization for multiple operating systems
  - Flexible memory management implementations
  - Multi-instruction atomic sequences
- Ultra-high performance Alpha implementation:
  - Dual-pipelined architecture
  - 150 MHz cycle time
  - Peak instruction execution of 300 million operations per second
- On-chip write buffer with four 32-byte entries
- Selectable data bus width and speed:
  - 64 or 128 bit data width
  - 75 MHz to 18.75 MHz bus speed
- On-chip pipelined floating point unit
- 8K byte data cache
- 8K byte instruction cache
- External cache memory support:
  - On-chip external secondary cache control
  - Programmable cache size and speed
- On-chip demand paged memory management unit:
  - 12 entry I-stream TB with 8 entries for 8K byte pages and 4 entries for 4M byte pages
  - 32 entry D-stream TB with each entry able to map 8K, 64K, 512K, or 4M byte pages
- On-chip parity and ECC generators and checkers
- Internal clock generator provides:
  - High-speed chip clock
  - Pair of programmable system clocks (CPU/2 to CPU/8)
- Programmable on-chip performance counters measure CPU and system performance
- Chip and module level test support
- 3.3-volt supply voltage
  - Lower power
  - Higher reliability
  - Interface to 5-volt logic

Description

Digital's 21064-AA microprocessor is the first in a family of chips to implement Digital's Alpha architecture. The 21064-AA microprocessor is a .75 micron CMOS based super-scalar super-pipelined processor using dual instruction issue and a 150 MHz cycle time. The Alpha architecture is a 64-bit RISC architecture designed with particular emphasis on speed, multiple instruction issue, multiple processors, and software migration from VAX/VMS and MIPS/ULTRIX.

21064-AA MicroArchitecture

Digital's 21064-AA microprocessor consists of four independent functional units: the integer execution unit (Ebox), floating point unit (Fbox), the load/store or address unit (Abox) and the branch unit. Other sections include the central control unit (Ibox) and the I and D cache.

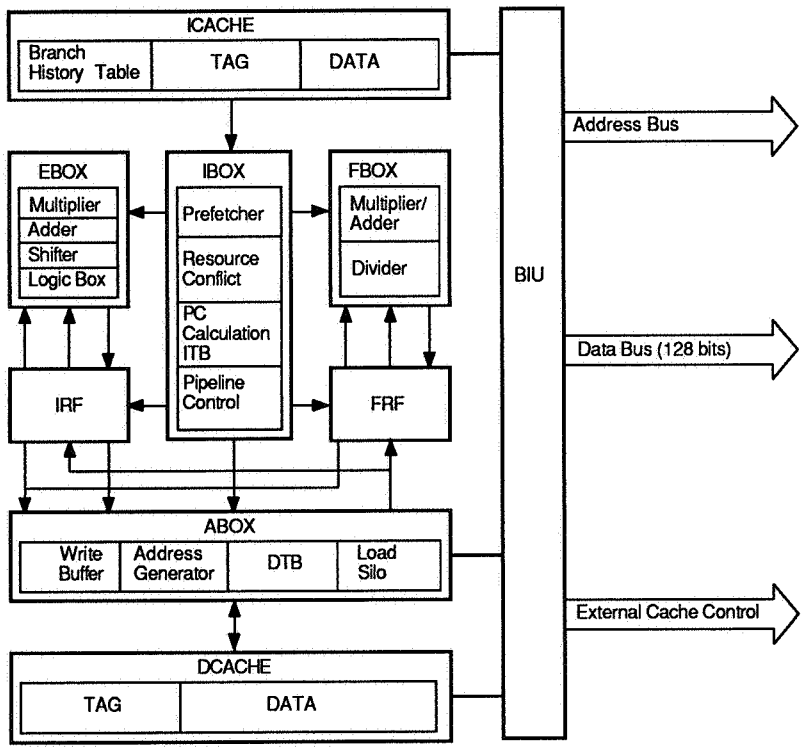
Ebox - Contains a 64-bit fully pipelined integer execution data path including: adder, logic box, barrel shifter, byte extract and mask, and independent integer multiplier. The Ebox also contains a 32-entry 64-bit integer register file.

Fbox - Contains a fully pipelined floating point unit and independent divider, supporting both IEEE and VAX floating point data types. IEEE single precision and double precision floating point data types are supported. VAX F_floating and G_floating data types are fully supported with limited support for the D_floating data type.

Abox - Contains five major sections: address translation data path, load silo, write buffer, data cache (Dcache) interface, and the external bus interface unit (BIU).

The Abox supports all integer and floating point load and store instructions, including address calculation and translation, and cache control logic.

Ibox - Performs instruction fetch, resource checks, and dual instruction issue to the Ebox, Abox, Fbox, or branch unit. In addition, the Ibox controls pipeline stalls, aborts and restarts.



PRELIMINARY

Pipeline Organization

The 21064-AA microprocessor uses a seven stage pipeline for integer operate and memory reference instructions, and a ten stage pipeline for floating point operate instructions. The Ibox maintains state for all pipeline stages to track outstanding register writes.

Cache Organization

The 21064-AA microprocessor contains two on-chip caches, data cache (Dcache) and instruction cache (Icache). The chip also supports an external cache.

Dcache - Contains 8K bytes and is a write through, direct mapped, read-allocate physical cache with 32-byte blocks.

Icache - Contains 8K bytes and is a physical direct-mapped cache with 32-byte blocks.

External Cache - The 21064-AA chip supports external cache built from off-the-shelf static RAMs. The 21064-AA chip directly controls the RAMs using its programmable external cache interface, allowing each implementation to make its own external cache speed and configuration trade-offs.

The external cache interface supports cache sizes from 0 to 8M bytes and a range of operating speeds which are sub-multiples of the chip clock.

Virtual Address Space

The virtual address is a 64-bit unsigned integer that specifies a byte location within the virtual address space. The 21064-AA microprocessor checks all 64-bits of a virtual address and implements a 43-bit subset of the address space. The 21064-AA supports a physical address space of 16G bytes.

Characteristics

Power Supply	Vss 0.0 V, Vdd 3.3 V ±5%
Operating Temperature	Tj max = 85°C
Storage Temperature Range	-55°C to 125°C
Power Dissipation @Vdd = 3.45V	23 W typical, 27.5 W maximum
Speed = 6.6 ns	

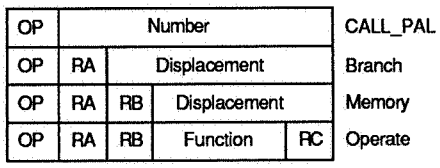
Alpha Architecture Summary

The 21064-AA microprocessor implements the Alpha architecture. The Alpha architecture supports:

- A fixed 32-bit instruction size
- Separate integer and floating point registers
  - 32 64-bit integer registers
  - 32 64-bit floating point registers
- 32-bit (longword) and 64-bit (quadword) integer along with 32-bit and 64-bit IEEE and VAX floating-point data types
- Memory access using a 64-bit virtual byte address
- Privileged Architecture Library Code (PALcode)

Instruction Set

Alpha instructions are all 32 bits in length using four different instruction formats specifying 0, 1, 2, or 3 5-bit register fields. Each format uses a 6-bit opcode.



CALL_PAL Instructions - vector to a privileged library of software that atomically performs both privileged and unprivileged functions.

Branch Instructions - Conditional branch instructions test a register for positive/negative, zero/nonzero, or even/odd, and perform a PC relative branch. Unconditional branch instructions perform either a PC relative or absolute jump using an arbitrary 64-bit register value. They can update a destination register with a return address.

Load/Store Instructions - can move either 32-bit or 64-bit quantities. 8-bit and 16-bit load/store operations are supported through an extensive set of in-register byte manipulations.

Integer Operate Instructions - manipulate full 64-bit values, and include a full complement of arithmetic, compare, logical, and shift instructions. In addition there are three 32-bit integer operates: add, subtract, and multiply.

In addition to the operation of conventional RISC architectures, the Alpha architecture provides scaled add/subtract for quick subscript calculation, 128-bit multiply for division by a constant and multi-precision arithmetic, conditional moves for avoiding branches, and an extensive set of in-register byte manipulation instructions.

PRELIMINARY

Floating-Point Operate Instructions - include four complete sets of instructions for IEEE single, IEEE double, VAX F_floating and VAX G_floating arithmetic. In addition to arithmetic instructions there are also instructions for conversions between floating and integer values including the VAX D_floating data type.

Privileged Architecture Library Code

PALcode is a privileged library of software that atomically performs such functions as the dispatching and servicing of interrupts, exceptions, task switching, and additional privileged and unprivileged user instructions as specified by operating systems using the CALL_PAL instruction.

PALcode is the only method of performing some operations on the hardware. In addition to the entire Alpha instruction set, a set of implementation specific instructions is provided.

PALcode runs in an environment with privileges enabled, instruction stream mapping disabled, and interrupts disabled. Disabling memory mapping allows PALcode to support functions such as TB miss routines. Disabling interrupts allows the instruction stream to provide multi-instruction sequences as atomic operations.

Memory Management

The Alpha memory management architecture is designed to meet several goals:

- Provide a large address space for instructions and data
- Provide convenient and efficient sharing of instructions and data.
- Provide independent read and write access protection
- Provide flexibility through programmable PALcode support

Alpha Architecture Compared to Conventional RISC Architecture

The Alpha architecture is different from conventional RISC architectures in a number of ways:

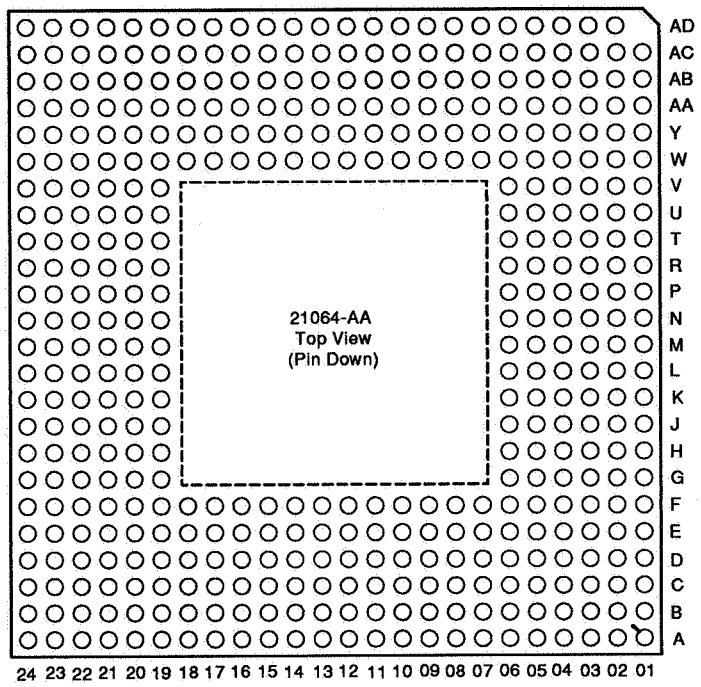
Feature	Difference
64-Bit Architecture	True 64-bit architecture with 64-bit data and address. Not a 32-bit architecture that was later expanded to 64 bits.
High Speed	The Alpha architecture was designed to allow very high-speed implementations. Simple instructions make it particularly easy to build implementations that issue multiple instructions every CPU cycle. There are no implementation specific pipeline timing hazards, no load delay slots, and no branch delay slots.
Multiprocessor Support	The Alpha architecture does not enforce strict read/write ordering between multiple processors. This allows multiprocessor implementations to easily use features such as: multi-bank caches, bypassed write buffers, write merging, and pipelined writes with retry on error. To maintain strict ordering between accesses as seen by a second processor, memory barrier instructions can be explicitly inserted in the program. The basic multi-processor interlocking primitive is a RISC style load_locked, modify, store_conditional sequence. If the sequence runs without interrupt, exception, or an interfering write from another processor, the store succeeds. Otherwise, the store fails and the program eventually must branch back and retry the sequence.
Multiple Operating Systems	The Alpha architecture provides flexibility by allowing the user to implement a privileged library of software for operating system specific operations. This allows Alpha to run full VMS using one version of this software library that mirrors many of the VAX operating system features, and to run OSF/1 using a different version that mirrors many of the MIPS operating system features. Additional operating system implementations can be efficiently supported.
Byte Manipulation	The Alpha architecture is unconventional in the approach to byte manipulation. Byte loads, stores, and operations are done with normal 64-bit instructions, crafted to keep the sequences short. Single-byte stores found in conventional RISC architectures force cache and memory implementations to include hardware byte operations and implement read-modify-write cycles which can complicate system design and reduce performance.
Arithmetic Traps	In contrast to conventional RISC architectures, the reporting of Alpha arithmetic traps (overflow, underflow, and others) are imprecise. This removes architectural bottlenecks that affect performance. If precise arithmetic exceptions are desired, trap barrier instructions can be explicitly inserted in the program to force traps to be delivered at specific points.
HINTS	Alpha includes a number of implementation specific HINTS aimed at allowing higher performance. Software is able to provide HINTS to the hardware that enable the hardware to optimize its operation. HINTS can help improve the utilization of the pipeline, cache memory, and translation lookaside buffers.

Signals

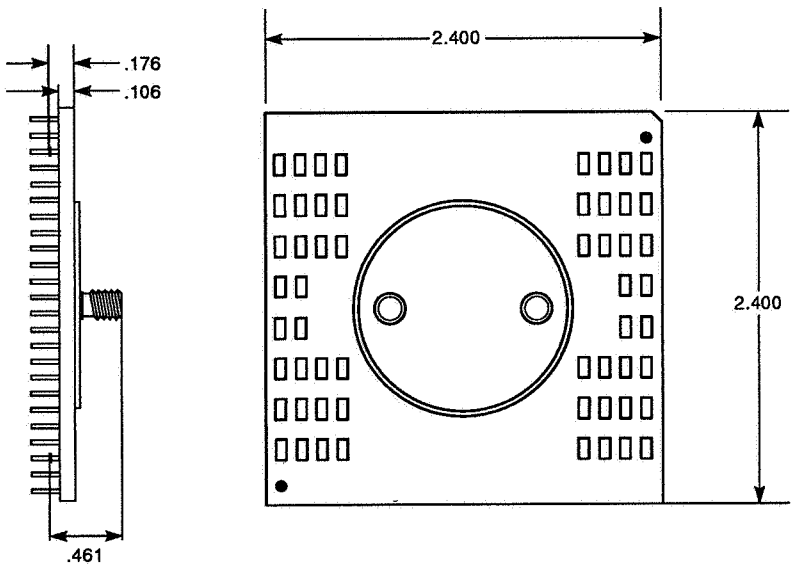
Name	Type	Function
adr_h 33:5	Input/Output	Address bus
data_h 127:0	Input/Output	Data bus
check_h 27:0	Input/Output	Check bit bus
dOE_l	Input	Data bus output enable
dWSel_h 1:0	Input	Data bus write data select
dRAck_h 2:0	Input	Data bus data acknowledge
tagCEOE_h	Output	External cache RAM tagCtl, tagAdr CE/OE
tagCtlWE_h	Output	External cache RAM tagCtl WE
tagCtlV_h	Input/Output	Tag valid
tagCtlS_h	Input/Output	Tag shared
tagCtlD_h	Input/Output	Tag dirty
tagCtlP_h	Input/Output	Tag V/S/D parity
tagAdr_h 33:17	Input	Tag address
tagAdrP_h	Input	Tag address parity
tagOK_h,_l	Input	Tag access from CPU is ok
tagEq_l	Output	Tag compare output
dataCEOE_h 3:0	Output	External cache RAM data CE/OE, longword
dataWE_h 3:0	Output	External cache RAM data WE, longword
dataA_h 4:3	Output	External cache RAM data A 4:3
holdReq_h	Input	Hold request
holdAck_h	Output	Hold acknowledge
cReq_h 2:0	Output	Cycle request
cWMask_h 7:0	Output	Cycle write mask
cAck_h 2:0	Input	Cycle acknowledge
iAdr_h 12:5	Input	Invalidate address, Dcache
dInvReq_h	Input	Invalidate request, Dcache
dMapWE_h	Output	External Dcache duplicate tag RAM WE
irq_h 5:0	Input	Interrupt request
sRomOE_l	Output	Serial ROM output enable
sRomD_h	Input	Serial ROM data/Rx data
sRomclk_h	Output	Serial ROM clock/Tx data
vRef	Input	Input reference
eclOut_h	Input	Output mode selection
perf_cnt_h 1:0	Input	Performance counter inputs
threestate_l	Input	Three state for testing
icMode_h 1:0	Input	Icache Test Mode Selection
cont_l	Input	Continuity for testing
clkIn_h,_l	Input	Clock input
testClkIn_h,_l	Input	Clock input for testing
cpuClkOut_h	Output	CPU clock output
sysClkOut1_h,_l	Output	System clock output, normal
sysClkOut2_h,_l	Output	System clock output, delayed
dcOk_h	Input	Power and clocks ok
reset_l	Input	Reset

Packaging

431 Pin Grid Array



Package Dimensions



PRELIMINARY

Information

For more information on Digital's 21064-AA Microprocessor call:

Voice: **1-800-DEC-2717**

TDD: **1-800-DEC-2515**

Orders may be placed through Digital's Technical OEM (TOEM) Sales Representatives. Call your local Digital Sales Office for details.

digital™

Copyright © Digital Equipment Corporation 1992

All Rights Reserved
Printed in U.S.A.

The following are trademarks of Digital Equipment Corporation: 21064, Digital, ULTRIX,VAX, VMS, and the Digital logo.

OSF and OSF/1 are trademarks of the Open Software Foundation, Inc.

PRELIMINARY

digital™

RECEIVED
- 5 DEC 1997

Digital Equipment Corporation and Intel Corporation Agreement

Digital Equipment Corporation and Intel Corporation have recently signed a comprehensive agreement to significantly enhance our commercial relationship and resolve our intellectual property issues.

Key provisions of the agreement, subject to US government approval, are:

- Intel will buy DIGITAL's semiconductor fabrication facilities and commit to a long term manufacturing agreement for Alpha microprocessors.

- DIGITAL will continue to invest in Alpha microprocessor development, fully supporting our long term road map and our Alpha customers.

- DIGITAL will design and manufacture Intel IA 64 based products in the future. Related to that:

    - DIGITAL will port our 64-bit UNIX to the IA 64 architecture to be available initially on Merced-based systems;
    - DIGITAL will work with Microsoft as it develops a common programming model for 64-bit Windows NT for the Alpha and IA 64 architectures.

DIGITAL and Intel have entered into a broad patent cross-licensing agreement This agreement:

- Assures a continuous supply of leading edge Alpha and Intel based systems.
- Leverages Intel's massive manufacturing capability.
- Promotes DIGITAL's existing Intel 32-bit based business.
- Maintains our long standing commitment to OpenVMS while significantly strengthening our UNIX and NT products and integration services.
- Strengthens DIGITAL's financial position.

DIGITAL's customers demand leading edge technology coupled with investment protection. This announcement allows us to better meet these demanding needs. We have strengthened our industry-leading 64-bit Alpha technology and provided long-term support for DIGITAL UNIX and Windows NT on both Alpha and IA 64 architectures. In addition, this agreement enriches our collaborative relationship with Intel for our current X-86 business. DIGITAL therefore is better able to provide you with the high performance, enterprise computing solutions you have come to expect.

If you require any further information, please contact your DIGITAL Account Manager on 01506-413241 or visit our website http://www.digital.com.

Digital Equipment Co Ltd
Lauder House
Almondvale Way
LIVINGSTON
West Lothian
EH54 6BX

Tel: 01506-413241
Fax: 01506-412459
E-mail: DigitalScotland@digital.com

Scotland Sales Manager - Tom Kelly

ALPHA - TECHNICAL CHARACTERISTICS

> Alpha is a multiple-instruction issue, pipelined, 64-bit, load/store, reduced instruction set architecture.

> It's the world's fastest IEEE compatible floating point chip.

> It is a dual-instruction processor, meaning that it can handle two instructions at once.

> It is a CMOS (Complimentary Metal Oxide Semiconductor) with 1.7m of transistors. Minimum feature size is 0.75 microns, transistor channel length is 0.5 microns and the chip operates at 3.3 volts.

> It handles 30 watts of power with 64-bit virtual and physical addresses and 64-bit integers and floating points, with no operating system or language bias and four billion times the addressable space of existing chips.

> With clock rates of up to 200MHz, the chip is capable of delivering 400 peak MIPs and 200 peak MFLOPS - over twice most of Digital's competitors.

> With a total transistor count of 1.68 million devices, the chip is a complete CPU, including full integer and floating-point execution units. These units, together with related addressing and branching units, are fully pipelined, and each is capable of launching a new operation every cycle.

> The chip includes two high speed memory caches. An eight Kbyte instruction cache provides two full 32-bit instructions per clock cycle to the instruction dispatch unit, and an eight Kbyte data cache can provide a 64-bit data access during each cycle. The resulting cache bandwidth of 3.2 Gigabytes/second far exceeds what could be accomplished if these cache units were not fully integrated.

> Initially, Alpha systems will be able to run OSF/1 and VMS - there is no bias towards a particular operating system or programming language in the architecture.

ALPHA PERFORMANCE & COMPETITIVE POSITIONING

Vendor	Digital	MIPS	Sun/TI	IBM	HP	Intel	Motorola
Device	21064	R4000	Viking	RIOS	PA-4	i860XP	88110
Max.Freq Internal	200 Mhz	100 Mhz	50 Mhz	50 Mhz	66 Mhz	50 Mhz	50 Mhz
No.Chips Required	1	1	1	7-9	2	1	1
Peak MIPS	400	100	150	200	132	150	150
Peak MFLOPS	200	50	50	100*	132*	100*	100
Base Arch Design	64-bit	64-bit	32-bit	32-bit	32-bit	32-bit	32-bit
Samples Available	Now	Now	N/A	N/A	N/A	Now	Mid-1992

* = combined fmul/fadd

Editor's Notes:

> Digital's CMOS technology runs faster than any other semiconductor manufacturer (including Intel, Motorola, IBM, TI, Cypress, NEC and Toshiba). CMOS-4 is the fourth generation of CMOS chip from Digital.

> Previous CMOS generations have delivered the world's fastest CISC microprocessors into Digital's products (for example VAX 6000 Model 500 with CMOS-3 chips operating at 62.5 MHz in 1990).

> CMOS-4 continues this world leading position in the VAX 6000 Model 600 with the 83MHz chip. This same technology is used to manufacture the Alpha CPU and a wide range of peripheral chips.

Author: COLETTE CREWS
Date: 11-Feb-1992
Posted-date: 24-Feb-1992
Subject: ALPHA FACT SHEET

Digital Equipment Corporation

Fact Sheet

PRODUCT NAME: 21064 150 MHz Microprocessor

OVERVIEW:

Digital's 21064 150 MHz microprocessor is the first in a family of chips to implement Digital's Alpha architecture. ("Alpha" is Digital's internal code name.) The 21064 microprocessor is a .75 micron CMOS-based super-scalar, super-pipelined processor using dual instruction issue and a 150 MHz cycle time (contact Digital for information on faster clock rate implementations). The Alpha architecture is a 64-bit RISC architecture designed with particular emphasis on speed, multiple instruction issue, and multiple processors.

21064 MICROPROCESSOR KEY FEATURES:

- o Implements the Advanced Alpha RISC Architecture
  - Optimized for high performance implementations
  - Multiprocessor support
  - IEEE single and double precision, VAX F_floating and G_floating, longword, and quadword data types
  - Cycle counter for code optimization
- o Single-chip implementation
- o 3.3-volt supply voltage
- o High performance
  - Dual-pipelined architecture
  - 150 MHz cycle time (contact Digital for information on faster clock rate implementations)
  - Peak instruction execution of 300 million operations per second (MIPS)
- o Privileged Architecture Library Code (PALcode) supports:
  - Optimization for multiple operating systems
  - Flexible memory management implementations
  - Multi-instruction atomic sequences
- o On-chip write buffer with four 32-byte entries
- o On-chip pipelined floating point unit

-more-

- On-chip 8 Kbyte data cache
- On-chip 8 Kbyte instruction cache
- On-chip demand paged memory management unit
  - 12-entry I-stream TB with 8 entries for 8 Kbyte pages and 4 entries for 4 Mbyte pages
  - 32-entry D-stream TB with each entry able to map 8 Kbyte, 64 Kbyte, 512 Kbyte, or 4 Mbyte pages
- On-chip parity and ECC generators and checkers
- On-chip internal clock generator provides:
  - High-speed chip clock
  - Pair of programmable system clocks (CPU/2 to CPU/8)
- Programmable on-chip performance counters measure CPU and system performance
- Selectable data bus width speed, 64 or 128 bits, 75 MHz to 18.75 MHz
- External cache memory support:
  - On-chip external secondary cache control
  - Programmable cache size and speed
- Chip and module level test support

Operating Characteristics:

Power Supply	Vss 0.0 V, Vdd 3.3 V +/-5%
Operating Temperature	Tj max = 85 deg. C
Storage Temperature Range	-55 deg. C to 125 deg. C
Power Dissipation @Vdd = 3.45V	23 W typical, 27.5 W maximum Speed = 6.6 ns
Die Size	13.9mm x 16.8mm
Transistor Count	1.68 million
Package	431 pin PGA

Alpha Architecture Summary:

The 21064 microprocessor implements the Alpha architecture. The Alpha architecture supports:

- A 64-bit virtual address space

-more-

- Separate integer and floating point registers
  - 32 64-bit integer registers
  - 32 64-bit floating point registers

- 32-bit (longword) and 64-bit (quadword) integer along with 32-bit and 64-bit IEEE and VAX floating-point data types

- Privileged Architecture Library Code (PALcode)

Instruction Set: Alpha instructions are all 32 bits in length using four different instruction formats specifying 0, 1, 2, or 3 five-bit register fields. Each format uses a 6-bit opcode.

Conditional branch instructions test a register for positive/negative, zero/nonzero, or even/odd, and perform a PC relative branch. Unconditional branch instructions perform either a PC relative or absolute jump using an arbitrary 64-bit register value. They can update a destination register with a return address.

Load/Store Instructions can move either 32-bit or 64-bit quantities. 8-bit and 16-bit load/store operations are supported through an extensive set of in-register byte manipulations.

Integer Operate Instructions manipulate full 64-bit values, and include a full complement of arithmetic, compare, logical, and shift instructions. In addition there are three 32-bit integer operates: add, subtract, and multiply.

The Alpha architecture provides scaled add/subtract for quick subscript calculation, 128-bit multiply for division by a constant and multiprecision arithmetic, conditional moves for avoiding branches, and an extensive set of in-register byte manipulation instructions.

Floating-Point Operate Instructions include four complete sets of instructions for IEEE single, IEEE double, VAX F_floating and VAX G_floating arithmetic. In addition to arithmetic instructions there are also instructions for conversions between floating and integer values including the VAX D_floating data type.

-more-

Privileged Architecture Library Code: PALcode is a privileged library of software that atomically performs such functions as the dispatching and servicing of interrupts, exceptions, task switching, and additional privileged and unprivileged user instructions as specified by operating systems using the CALL_PAL instruction.

PALcode is the only method of performing some operations on the hardware. In addition to the entire Alpha instruction set, a set of implementation specific instructions is provided.

PALcode runs in an environment with privileges enabled, instruction stream mapping disabled, and interrupts disabled. Disabling memory mapping allows PALcode to support functions such as TB miss routines. Disabling interrupts allows the instruction stream to provide multi-instruction sequences as atomic operations.

Memory Management: The Alpha memory management architecture is designed to provide a large address space for instructions and data, convenient and efficient sharing of instructions and data, independent read and write access protection, and flexibility through programmable PALcode support.

Microarchitecture:

Digital's 21064 microprocessor consists of four independent functional units: the integer execution unit (Ebox), floating point unit (Fbox), the load/store or address unit (Abox), and the branch unit. Other sections include the central control unit (Ibox) and the I and D cache.

The Ebox contains a 64-bit, fully pipelined integer execution data path including: adder, logic box, barrel shifter, byte extract and mask, and independent integer multiplier. The Ebox also contains a 32-entry 64-bit integer register file.

The Fbox contains a fully pipelined floating point unit and independent divider, supporting both IEEE and VAX floating point data types. IEEE single-precision and double-precision floating point data types are supported. VAX F_floating and G_floating data types are fully supported with limited support for the D_floating data type.

-more-

The Abox contains five major sections: address translation data path, load silo, write buffer, Data cache (Dcache), and the external bus interface unit (BIU). The Abox supports all integer and floating point load and store instructions, including address calculation and translation, and cache control logic.

The Ibox performs instruction fetch, resource checks, and dual instruction issue to the Ebox, Abox, Fbox, or branch unit. In addition, the Ibox controls pipeline stalls, aborts, and restarts.

Pipeline Organization: The 21064 microprocessor uses a seven-stage pipeline for integer operate and memory reference instructions, and a ten-stage pipeline for floating point operate instructions. The Ibox maintains state for all pipeline stages to track outstanding register writes.

Cache Organization: The 21064 microprocessor contains two on-chip caches, data cache (D-cache) and instruction cache (I-cache). The chip also supports an external cache. The D-cache contains 8 Kbytes and is a write-through, direct-mapped, read-allocate physical cache with 32-byte blocks. The I-cache contains 8 Kbytes and is a physical direct-mapped cache with 32-byte blocks.

The 21064 chip supports external cache built from off-the-shelf static RAMs. The 21064 chip directly controls the RAMs using its programmable external cache interface, allowing each implementation to make its own external cache speed and configuration trade-offs.

The external cache interface supports cache sizes from 0 to 8 Mbytes and a range of operating speeds that are sub-multiples of the chip clock.

Virtual Address Space: The virtual address is a 64-bit unsigned integer that specifies a byte location within the virtual address space. The 21064 microprocessor checks all 64 bits of a virtual address and implements a 43-bit subset of the address space. The 21064 supports a physical address space of 16 Gbytes.

-more-

PRICING: Samples (through June 1992) $3375 each. Volume quantities (July 1992) starting at $1650 (1,000 pieces).

AVAILABILITY: Sample parts available February 1992 for customer evaluation. Quantities to support customers' volume ramps available in July 1992.

# # #

For Further Information Contact our Hotline:

1-800-DEC-2717
1-800-DEC-2515 (Telecommunications Device for the Deaf TDD)
508-568-6868 (local number)

Note to Editors: DEC, the Digital logo, VAX and VMS are trademarks of Digital Equipment Corporation.

OSF/1 is a registered trademark of Open Software Foundation, Inc.

UNIX is a registered trademark of UNIX System Laboratories, Inc.

PAPER TA6.2

A 200MHz 64 Bit Dual Issue CMOS Microprocessor

by

Daniel Dobberpuhl, Richard Witek, Randy Allmon, Robert Anglin, Sharon Britton, Linda Chao, Robert Conrad, Daniel Dever, Bruce Gieseke, Gregory Hoeppner, John Kowaleski, Kathryn Kuchler, Maureen Ladd, Michael Leary, Liam Madden, Edward McLellan, Derrick Meyer, James Montanaro, Donald Priore, Vidya Rajagopalan, Sridhar Samudrala, Sribalan Santhanam

DIGITAL EQUIPMENT CORPORATION

77 Reed Road

Hudson, MA 01749

PRESENTER and CONTACT:

Dan Dobberpuhl, Senior Consulting Engineer
Digital Equipment Corp., HLO2-3/J3
Hudson, MA 01749
tele: (508) 568 4469
fax: (508) 568 4681

A 200MHz 64 Bit Dual Issue CMOS Microprocessor

by

Daniel Dobberpuhl, Richard Witek, Randy Allmon, Robert Anglin, Sharon Britton, Linda Chao, Robert Conrad, Daniel Dever, Bruce Gieseke, Gregory Hoeppner, John Kowaleski, Kathryn Kuchler, Maureen Ladd, Michael Leary, Liam Madden, Edward McLellan, Derrick Meyer, James Montanaro, Donald Priore, Vidya Rajagopalan, Sridhar Samudrala, Sribalan Santhanam

Semiconductor Engineering Group
DIGITAL EQUIPMENT CORPORATION
Hudson, MA

A RISC-style microprocessor has been designed and tested which operates up to 200MHz. The chip implements a new 64 bit architecture, designed to provide a huge linear address space and to be devoid of bottlenecks which would impede highly concurrent implementations. Fully pipelined and capable of issuing two instructions per clock cycle, this implementation can execute up to 400 million operations per second. The chip includes an 8KB I-Cache, 8KB D-Cache and two associated translation buffers, a four entry 32 byte/entry write buffer, a pipelined 64b integer execution unit with 32 entry register file, and a pipelined floating point unit with an additional 32 registers. The pin interface includes integral support for an external secondary cache. The package is a 431 pin PGA with 140 pins dedicated to VDD/VSS. The chip is fabricated in 0.75u n-well CMOS with 3 layers of metallization (Figure 1). The die measures 16.8mm X 13.9mm and contains 1.68 million transistors. Power dissipation is 30 watts from a 3.3V supply at 200MHz. A photomicrograph is shown in Figure 2.

The architecture includes comprehensive support for both 32 and 64 bit operations on an instruction-specific basis. It is designed to provide continuity to an earlier CISC architecture without sacrificing RISC performance characteristics. All operate instructions are register to register while memory operations are strictly load/store.

The instruction issue scheme supports pair-wise execution among combinations of four basic units: Load/Store, Integer Operate, Floating Point Operate, and Branch. The pipeline depth is seven cycles for everything except floating point operate which is ten cycles. Only integer multiply and floating divide are not fully pipelined. Integer latency is generally one cycle, load latency is three cycles, and floating point latency is six cycles. Instructions are issued in order and under the control of register scoreboarding logic. The scoreboarding logic also controls result bypassing among the various units. The issue point is at pipeline stage four, beyond which the pipeline does not stall.

The floating point unit is a fully pipelined 64b floating-point processor which supports both VAX standard and IEEE standard data types and roundings. It is capable of generating a 64b result every cycle for all operations except divide. The pipeline stages include a 64b adder, a leading one detector using input operands, two parallel exponent adders in stage 1, a radix-8 pipelined Booth multiplier organized as 8 odd and 8 even rows, a 64b shifter capable of shifting both left and right by up to 63 bits, and a 64b double adder to enable parallel addition and rounding in the final stage.

Many novel circuit structures and detailed analysis techniques were developed to support the clock rate in conjunction with the complexity demanded by the concurrence and wide data paths. The clocking method is level sensitive, single phase. Since there is no "dead time" with this scheme, it is imperative that clock integrity be assured to avoid race-through in latches. It is also important to avoid clock skew contribution to delay paths. We chose to use a single driver approach to clock distribution, using the third metal layer to do the majority of clock routing (as well as VDD/VSS). Total capacitive load on the clock driver is 3250pF, requiring a final driver width of 250Ku (10 inches) and 100Ku (4 inches) for PMOS and NMOS respectively. The clock driver resides in the horizontal center of the chip and extends vertically from top to bottom of the core. A method to extract and display clock skew was developed to analyze the grid. Figure 3 shows a topographic representation of on-chip clock skew.

A difficult circuit problem was the 64b adder portion of the integer and floating point ALUs. Unlike a previous high speed design [1], we set a goal to achieve single cycle latency in this unit. To do this, a combination of logic and circuit techniques was used. The logical scheme (Figure 4) is a hybrid of two techniques - Manchester carries for the initial 8 bit groups, followed by a logarithmic carry select tree [2]. The Manchester scheme is unique in that the NMOS chain is precharged low and is conditionally pulled high. This avoids threshold delays in the pass transistors and was found to improve performance of the carry chain by 10% over the pull-down approach.

To provide maximum flexibility in applications, the external interface allows for several different modes of operation. This includes choice of logic family (CMOS/TTL or ECL) as well as bus width (64/128b), external cache size and access time, and BIU clock rate. These parameters are set into mode registers during chip power-up. The logic family choice provided an interesting circuit challenge. The input receivers are differential amplifiers that utilize an external reference level. To maintain signal integrity of this reference voltage, it is resistively isolated and RC filtered at each receiver as shown in Figure 5. The output driver presented a more difficult problem due to the 3.3V VDD chip power supply. To provide a good interface to ECL, it is important that the output driver pull to the VDD rail (for ECL operation VDD=0V, VSS=-3.3V). This precludes using NMOS pull- ups. PMOS pull-ups have the problem of well junction forward bias and PMOS turn-on when bi-directional outputs are connected to 5V logic in CMOS/TTL mode. The solution, as shown in Figure 6, is a unique floating well driver circuit which avoids the cost of series PMOS pull-ups in the final stage [3].

Acknowledgments:

The authors acknowledge the contributions of the following individuals : H. Akhiani, B. Benschneider, L. Binder, D. Bertucci, V. Calderon, S. Carroll, G. Cheney, D. Conroy, B. Cooper, M. D'Addeo, T. Daum, M. Decker, T. Equi, S. Hassoun, B. Hicks, C. Holub, T. Jacobs, A. Jain, D. Jessel, M. Kantrowitz, S. Kreider, S. Kumar, A. Ladd, D. Lau, S. Lloyd, S. Lowell, R. Matthew, S. Meier, S. Morris, J. Pan, V. Rai, N. Rethman, S. Shah, D. Sites, D. Sleeper, C. Somanathan, E. StJohn, M. Tareila, M. Taylor,

Lang Tran, Luan Tran, H. Tumblin, L. Wolf, Y. Yen.

References:

[1] Conrad, R., et. al., "A 50 MIPS (Peak) 32/64b Microprocessor", ISSCC DIGEST OF TECHNICAL PAPERS, pp76-77, Feb., 1989.

[2] Sklansky, J., "Conditional-Sum Addition Logic", IRE Trans. Electron. Comput. EC-9:pp226-231, 1960.

[3] Lee, H., et. al., "An Experimental 1Mb CMOS SRAM with Configurable Organization and Operation", ISSCC DIGEST OF TECHNICAL PAPERS, pp180-181, Feb., 1988.

FIGURE TITLES

Figure 1: CMOS-4 Technology

Figure 2: Chip Micrograph - 200MHz Microprocessor

Figure 3: Clock Delay vs Position

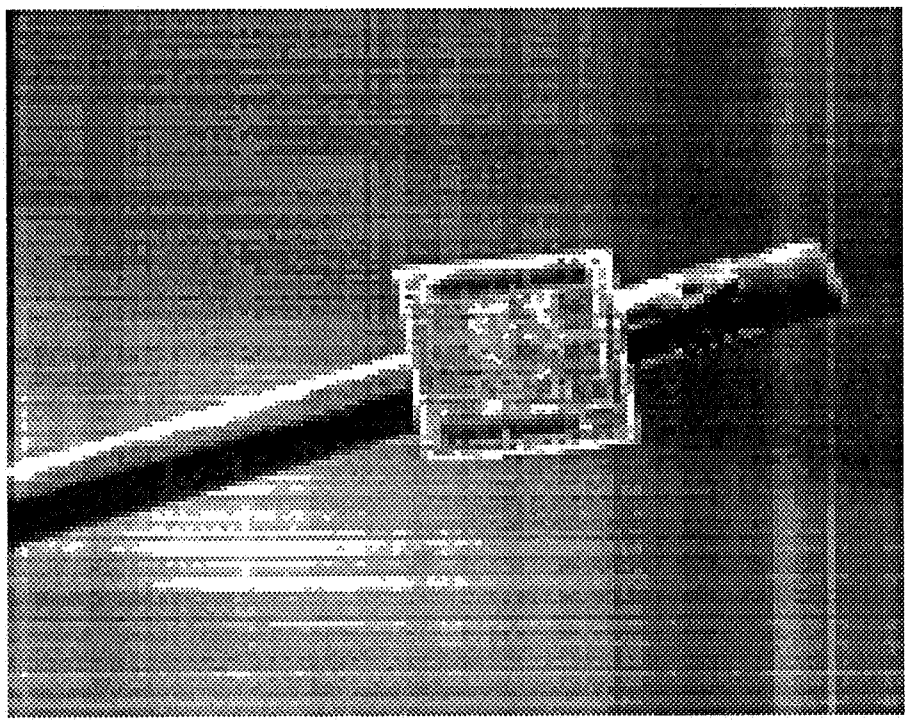
Figure 4: Diagram of 64 Bit Adder

Figure 5: Reference Voltage Filter Network at Each Input Pin

Figure 6: Floating Well Output Driver

Digital's 21064 Microprocessor

21st Century 64-bit RISC Computing Architecture—Today!



The Highest Performance Microprocessor in the Industry

The 21064 microprocessor is the first product implementing Digital's new 64-bit RISC computing architecture. Digital's internal code name for this architecture is Alpha.

The result of a twelve-year investment in research, design, and fabrication, Digital's 21064 microprocessor has the highest performance of any microprocessor in the industry. The current 150 MHz version offers performance that peaks at 300 million instructions per second (MIPS) and 150 million floating-point operations per second (MFLOPS). Computer systems built around this microprocessor will be part of a new generation—a generation of systems that is setting the standard for computing in the 21st century.

The 21064 microprocessor is the first implementation of an architecture designed with the capacity and performance potential to last at least 25 years. In keeping with the growing needs of users everywhere, Digital designed the architecture so that future implementations can realize this potential through increased chip speed, advances in multiple instruction issue, and multiprocessor configurations.

With the 21064 microprocessor, everyone wins. *Current Digital customers* know that the applications they use today will migrate easily and cost-effectively to the new architecture. *Software developers* will find that the software they wrote for VAX and DECsystem computers can migrate easily to 21064-based systems. Any software they design to take advantage of the Alpha architecture will have a variety of platforms that will provide industry-leading performance into the 21st century. And *hardware developers* will win when they create products that use the 21064 microprocessor.

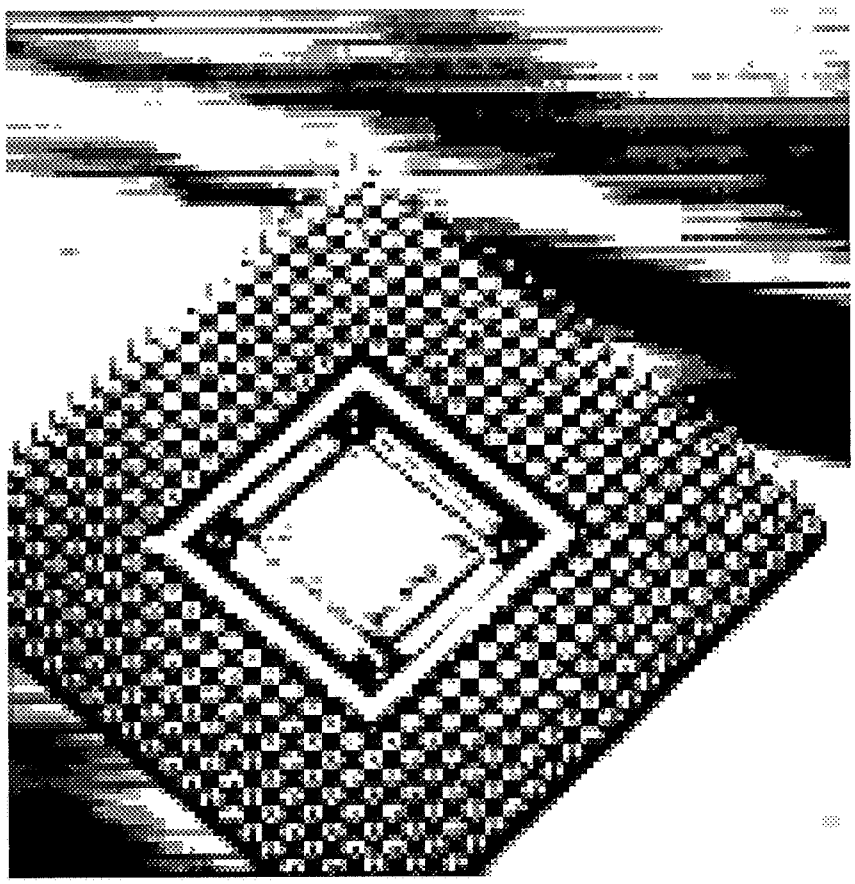
Highlights

- The 21064 microprocessor implements the full 64-bit Alpha architecture using Digital's state-of-the-art CMOS technology.
- The current 150 MHz implementation provides the highest CPU performance in the industry — 300 peak MIPS and 150 peak MFLOPS.
- Its single-chip implementation provides higher performance, improved reliability, and significantly lower cost.
- Full application and development support is available.

Alpha — An Architecture for the 21st Century

The Alpha architecture is a full RISC architecture without compromises in its short- or long-term performance. This is not a 32-bit architecture expanded to 64 bits. Alpha was designed to be a 64-bit architecture right from the very start.

Why 64 bits? A full 64-bit system gives you an extended address space for truly demanding applications. This is particularly important for applications such as molecular modeling, advanced CAD, and weather forecasting — which are expected to reach performance limitations with 32-bit technology in the near future.



Designed for Speed

The initial chip implementation is based on Digital's leadership 0.75 micron CMOS VLSI technology, which has produced the fastest chip in the industry. This technology is the result of more than twelve continuous years of investment in quality, design, and process improvements. The result: the first-pass chips were fully functional and worked at full speed — a stunning achievement considering the complexity involved in integrating 1.7 million discrete devices.

When you look at the 21064 microprocessor, you are looking at a technology that will continue to improve as the years go by. Digital's long-term investment in semiconductor technology ensures ongoing performance improvements. The Alpha architecture has been designed so that it can cater not only to increased multiple instruction-issue implementations, but also to the needs of massively parallel processing (MPP).

One of the reasons the 21064 microprocessor is so fast is that it is a single-chip implementation of the architecture. In addition to the performance edge, single-chip implementations offer other advantages that make them cheaper to produce than multiple-chip implementations. They use less power, generate less heat, and require less space on the circuit board. Also, fewer components usually mean a more reliable product.

The 21064 is a super-scalar, super-pipelined implementation of the Alpha architecture. Super-pipelined means that an instruction is issued to the functional units at every clock tick and the results are pipelined. Being super-scalar, the architecture allows the instruction unit to issue two instructions per clock tick, resulting in significantly higher throughput and performance.

The Founding Member of a Large Extended Family

The 21064 is the first in a family of microprocessor implementations of the Alpha architecture. This family of devices will provide optimized solutions for systems from the desktop to the datacenter. This is unlike many competitors' RISC implementations, which are oriented to only one style of system, such as workstations. Like the VAX family, all implementations of the Alpha architecture will be capable of running the same system and application software.

Multiprocessing Supports the Biggest Workloads

The 21064 has been designed to support multiple-processor configurations. When processing demands are especially high, multiple processors can be linked to share massive workloads. With multiprocessing, systems implementing the 21064 can range from single-chip computers to massively parallel processor systems. As years of success with multiprocessing on Digital's VAX systems have proven, this kind of flexibility is a must for high-performance systems.

Choose the Best Operating System for the Job

Several characteristics of the Alpha architecture set new standards for comparison in the industry. The Alpha architecture has been designed from the start to be unbiased towards operating systems. For example, the architecture is being introduced to support both the OSF/1 and VMS operating systems — with few compromises in speed or functionality.

The architecture is new, but users' investments in VAX and DECsystem applications is protected with technology like the binary translator, which allows existing applications to run on 21064-based systems. The binary translator technology is state-of-the-art and is not limited to only VAX or DECsystem computers — other translators can be built.

The 21064 microprocessor also includes both VAX and IEEE floating data types. This is consistent with the VAX architecture and supports the most widely accepted industry standards.

The architecture includes a very flexible, privileged library of software for optimizing the performance of specific operating systems. One version of this software allows the architecture to run a full version of the VMS operating system that mirrors many of the VAX operating system features. Another version of the software supports a version of the OSF/1 operating system that mirrors all of the DECsystem operating system features. Other versions could be tailored for real-time, teaching, or other special uses. This library makes it possible for virtually any operating system to run efficiently on Alpha systems.

The Open Advantage

The 21064 microprocessor is strong testimony to Digital's commitment to delivering the Open Advantage to our customers. Openness also requires open business practices. So, in addition to selling systems built around the 21064 microprocessor, Digital will license the technology and form partnerships with vendors who want to use the architecture in their systems or embed the microprocessor in special-purpose products. To complement the open technology and open business practices, open service offerings range from simple requirements such as replacing spare Digital or non-Digital parts to actually outsourcing and managing complex information centers.

digital™

Specifications

Process Technology	.75 micron CMOS
Cycle Time	150 MHz (6.6 ns)
Die Size	13.9mm x 16.8mm
Transistor Count	1.68 million
Package	431 pin PGA
Number of Signal Pins	291
Power Dissipation	23 W at 6.6 ns cycle
Power Supply	3.3 volts
Clocking Input	300 MHz differential
On-chip D-cache	8 Kbyte, physical, direct-mapped, write-through, 32-byte line, 32-byte fill
On-chip I-cache	8 Kbyte, physical, direct-mapped, 32-byte line, 32-byte fill, 64 ASNs
On-chip DTB	32-entry; fully-associative; 8-Kbyte, 64-Kbyte, 256-Kbyte, 4-Mbyte page sizes
On-chip ITB	8-entry, fully associative, 8-Kbyte page plus 4-entry, fully-associative, 4-Mbyte page
Floating Point Unit	On-chip FPU supports both IEEE and VAX floating point
Bus	Separate data and address bus; 128-bit/64-bit data bus
Serial ROM Interface	Allows the chip to directly access serial ROM
Virtual Address Size	64 bits checked; 43 bits implemented
Physical Address Size	34 bits implemented
Page Size	8 Kbytes
Issue Rate	2 instructions per cycle to A-box, E-box, or F-box
Integer Pipeline	7-stage pipeline
Floating Pipeline	10-stage pipeline

Service and Support

Twenty-four hours a day, 365 days a year, Digital's Customer Support Centers around the world offer state-of-the-art hardware, software, and network solutions. Besides supporting our own equipment, Digital provides first-class service for other vendors' equipment.

More than 40,000 professionals in 450 locations worldwide deliver a wide range of services. These include services for developing information systems solutions and integrating them with your business, organization, technical environment, and architecture.

Digital is committed to providing a wide range of services designed specifically for vendors who are incorporating the 21064 microprocessor into their own products. Services range from training, telephone support, and design-in services, to service partnerships and private-label service offerings for vendors' end customers.

For More Information

To learn more about pricing and availability of the 21064 microprocessor in its 150 MHz or faster clock rate versions, contact your local Digital sales representative. Or, in the United States, dial 1-800-DEC-2717; 1-800-DEC-2515 TTY.

Digital believes the information in this publication is accurate as of its publication date; such information is subject to change without notice.

The following are trademarks of Digital Equipment Corporation: DEC, DECsystem, the DIGITAL logo, Open Advantage, VAX, and VMS.

OSF/1 is a registered trademark of Open Software Foundation, Inc.

EC-F1720-10 Copyright© 1992 Digital Equipment Corporation. All Rights Reserved.

Author: COLETTE CREWS
Date: 13-Feb-1992
Posted-date: 21-Feb-1992
Precedence: 1
Subject: ALPHA BACKGROUNDER

Barbara Watterson
(508) 568-6501

Backgrounder for Editors

ALPHA:

A TECHNOLOGY FOUNDATION FOR THE 21ST CENTURY

As the world approaches the 21st century, powerful factors are driving the industry toward new computing architectures. Power-hungry applications, such as molecular modeling, econometric forecasting, visualization, and imaging, are gobbling up compute cycles and storage space at a rapidly accelerating pace. Global networking and open computing are changing the ground rules for communications and integration of applications across geographies. At the same time, technologies are changing much more rapidly and unpredictably than they were even five years ago.

In this environment, organizations have to look for more than evolutionary improvements within existing platforms to achieve their long-term goals and emerge as winners in tomorrow's markets. They need new technologies and computing architectures that are rich enough to deliver the performance and solution choices they require for the future, while protecting their investment in current technologies.

Digital continually evaluates the needs of its customers and the market--for today and the future. During this process, Digital has identified key characteristics that are essential for an optimal computing architecture for the future.

By the turn of the century organizations will find that a 64-bit architecture is essential. As computing demands increase, today's 32-bit architectures will limit current computing systems. Based on an historical "consumption" of 6/10ths of a bit per year, existing 32-bit architectures will soon run out of address space. However, moving from 32-bit to 64-bit addressing has a significant impact on software compatibility and represents a fundamental architecture change.

-more-

In existing architectures, performance improvements are derived primarily from increases in clock speed. The future will demand more. Computing performance in the industry has improved by a factor of 100 over the past ten years. Given the ever-accelerating rate of technical advancement and demand for performance, it is likely that an architecture with a 25-year horizon will need to scale over a performance range of at least 1,000. Clock speed increases will not be enough. Processors will have to issue multiple instructions at the same time and multiple chips will have to be strung together to share the work. New architectures will be needed to optimize these capabilities. Architectures of the future will also need the ability to run any operating system and any language. Today's architectures are optimized for only one or two languages and operating systems.

Speed barriers in today's architectures will also have to be eliminated. Most current architectures haven't anticipated the dramatic increase in chip speeds, especially in RISC chips. Inherent bottlenecks result, precluding efficient use of new technologies. In addition, customers will want performance based on world-class fabrication technologies. They will seek out architectures that are implemented in chips built by leading microprocessor manufacturers.

Alpha, Digital's new computing architecture for the 21st century, incorporates all of these essential characteristics. It is designed to deliver lasting solutions that will endure over the next 25 years. Alpha computing products, based on this architecture, will help organizations manage their global operations, achieve reduced costs, comply with regulations, and share information with customers and suppliers. These products can help scientists, educators, health care professionals, engineers, and many others accomplish their work more easily, productively, and at a lower cost. Alpha's open approach will deliver solutions that encompass multivendor products in a global, distributed, and cost effective computing environment.

Alpha represents Digital's commitment to be the technology and solutions leader in open computing through the 1990s and beyond. Alpha is an internal code name for three things. First, it is Digital's new, open, 64-bit Reduced Instruction Set (RISC) computing architecture. This advanced architecture provides very high levels of performance and reliability. It is open, scalable, and designed to endure over a period of 25 years or more. Second, Alpha is a single-chip implementation of the new architecture, offering double the speed of any commercially available competing technology today. And third, Alpha will be a family of systems, enabling technologies, and services that span the desktop to the data center.

-more-

Investment protection is a key Alpha design feature. Through significant engineering efforts in hardware and software technologies, Digital will make it easy for customers--both VAX VMS and DECsystems OSF/1--to move smoothly to Alpha. This means that customers can satisfy their computing needs today from a wide range of Digital systems, and be assured of a clear and simple path to add Alpha systems to this environment as their needs change.

A NEW ARCHITECTURE

A computing architecture is a set of structural rules and interface standards for building computer systems with a similar look and feel. For example, IBM's System 360, Motorola's 68000, and Digital's PDP and VAX families of systems represent computing architectures. A computing architecture is important because it determines the limits of system performance and capacity, and is the basis for binary compatibility between products.

Digital's Alpha architecture is a robust, long-lasting architecture that provides both superior price performance and binary compatibility. It is a powerful foundation that subsequent layers of computing technology can leverage to satisfy changing customer requirements.

Digital's Alpha architecture incorporates flat 64-bit addressing that provides four billion times the address space of a 32-bit architecture. In addition to a virtually limitless address space, Alpha's high performance features, scalability, and "openness" make it the first architecture able to accommodate both technology and application changes through the next 25 years.

Digital has already demonstrated its ability to develop an architecture with long-term viability that spans multiple generations of semiconductor technology innovation. Digital's VAX systems have offered binary compatibility and high performance across a broad range of systems for 15 years, and will continue to do so for years to come. Alpha will follow Digital's industry-leading tradition of evolution based on architectural compatibility.

-more-

HIGH-SPEED PROCESSOR ON A CHIP

Digital's new RISC microprocessor, the 21064-AA, is the first product to implement the Alpha architecture. With a clock speed of 150 MegaHertz, it is the world's fastest microprocessor. By comparison, the fastest RISC microprocessors available today are approaching only 100 MegaHertz. The first chip-level design of the Alpha architecture has demonstrated performance at 200 MHz, and Digital plans to offer versions of the chip at different speeds over time.

The Alpha 21064-AA is a dual-issue processor--able to launch two instructions at once. It processes 64-bit virtual and physical addresses and 64-bit integers and floating point numbers.

Digital manufactures the Alpha 21064-AA in state-of-the-art facilities in Hudson, Massachusetts and South Queensferry, Scotland using the company's fourth generation of complementary metal oxide semiconductor chip technology (CMOS-4). This advanced process technology is tuned to very high-speed complex functions with high-speed on-chip memory, distinguishing it from the manufacturing processes of merchant semiconductor manufacturers. In addition, the CMOS-4 process produces chips with very high reliability. Because Digital 21064-AA chips are produced to run at 3.3 volts compared to the 5.0 volts common throughout the industry, they use less power and run cooler, making them more reliable than microprocessors of comparable speed. Also, single-chip implementations such as the Digital 21064-AA microprocessor, are inherently more reliable than multiple-chip processors.

Alpha's built-in scalability will be able to accommodate products from palmtop systems to supercomputers. Alpha is designed for a 1,000 times performance improvement--up to 400 billion instructions per second--during its lifetime. It can work as a single chip at the low end or with hundreds or thousands of chips in a massively parallel processing environment at the high end.

-more-

AN ENDURING TECHNOLOGY

In the future, organizations will increase their demands for open computing and integration of products from multiple vendors. At the same time, suppliers in the computing industry will require massive investments in core technologies to assure success. There will be a growing number of alliances and collaborations in many areas as suppliers realize that they can't do everything alone. The industry will continue to converge and only a few computing architectures will survive.

Digital has the leadership competency in core technologies and the resources to ensure that Alpha will endure. Digital's 35 years of experience in computer architecture and systems design made the Alpha architecture possible. Digital's semiconductor group, responsible for Alpha chip design and delivery, has a successful fifteen-year track record, and includes some of the industry's most experienced chip design and fabrication experts. In addition, Digital's extensive experience in networking, compilers, and enabling software, such as Network Application Support (NAS) and CASE tools, will contribute to a broad set of Alpha product offerings in the coming years.

To achieve the broadest possible use of the Alpha architecture with the widest range of operating environments, Digital is entering into new alliances. These will include licensing agreements with semiconductor manufacturers for the Alpha architecture as well as relationships with computer companies who will use Alpha chips in their systems. With Alpha, Digital will become a "merchant" company, selling chips in volume to the outside world. Digital is already working with Cray Research who will use Alpha chips in its new MPP supercomputer and Kubota Corporation who will use Alpha chips in building new, high-performance graphics workstations. Digital anticipates that others will soon take advantage of the Alpha technology.

Digital is also marketing the Alpha chip for use in embedded, technical original equipment manufacturer (OEM) applications. To make a full range of applications, tailored to Alpha's strengths, available to customers, Digital is also working closely with software application developers on a support program to help software companies move their applications to Alpha.

-more-

A MIGRATION STRATEGY FOR INVESTMENT PROTECTION

Ensuring that Alpha products integrate seamlessly with other Digital platforms has been an Alpha design goal since its inception. Digital's migration strategy for Alpha will give customers, who own or buy a system from Digital today, a clear and simple path to add Alpha systems, as they become available over time, to their existing computing environments. Customer investments in applications and data, training, and peripherals will be preserved. No other vendor has provided this level of investment protection with its RISC offerings.

Built in data compatibility will allow customers to move information back and forth between VAX VMS and Alpha VMS systems. In addition, VAX VMS systems and Alpha systems will be able to share disks in a VAXcluster. Customers will be able to move information back and forth between DECsystems/DECstations running OSF/1 and Alpha systems running OSF/1 through similar data compatibility.

Source code compatibility will enable customers to run most VAX VMS programs on Alpha VMS systems following a simple re-compile and re-link. This capability will empower Digital customers to redeploy existing applications on Alpha systems to accelerate performance. In addition, customers can use the VAX systems they buy today to develop applications to run on both VAX and Alpha systems of tomorrow. Similar source code compatibility will allow customers to run their DECsystems/DECstations OSF/1 applications on Alpha systems running OSF/1 following a re-compile and re-link.

Through sophisticated Digital binary translators, customers will also be able to run most DECsystem/DECstation OSF/1 and VAX VMS program images on Alpha systems. These software tools will translate OSF/1 executable images for DECsystems/DECstations and VMS executable images for VAX to the corresponding executable images on Alpha. Using these features, customers can run applications on Alpha that they can't or don't want to re-compile and achieve performance comparable to the then-current OSF/1 or VMS systems.

-more-

In addition, Digital customers will benefit from common user interfaces and support for common peripherals. For example, all Alpha systems will provide DECwindows and Motif, eliminating the need for retraining of users who are already familiar with these environments. To protect customers' investments in peripherals, Alpha will support the Turbochannel, XMI, CI, SCSI, DSSI, and Future+ buses. Customers will be able to move many VAX or DECsystem/DECstation peripherals to Alpha systems at a time that makes the most business sense for them. Because peripherals often represent 50% or more of system cost, customers will derive a significant financial benefit through this compatibility.

SERVICE AND SUPPORT

Even the best technology is useful only if it can be maintained and used effectively. Training, consulting support, and maintenance services are vital to maximizing investment in today's advanced technologies. Digital's Alpha Services deliver support designed specifically to meet the needs of Alpha customers.

Through its Alpha Vendor/Channels Services, Digital will provide comprehensive support for vendors who buy and incorporate Alpha products in their own product offerings. Digital professionals will be available to assist vendors in designing, prototyping, testing, manufacturing, distributing, and servicing Alpha products. Vendors can take advantage of tailored support packages, consulting, and training programs. For example, Digital can help vendors plan successful service programs and even provide services, on Digital and non-Digital products, on behalf of vendors. Currently Digital supports more than 200 software products from 50 vendors and over 10,000 products from 1,000 hardware vendors.

Alpha End-User Services will help end users in planning, designing, implementing, and managing an Alpha environment. Service offerings will include consulting, education and training, client/server management services, integration and migration services, and others.

-more-

IN SUMMARY

Alpha clearly positions Digital for the future. It is a technology foundation supporting open computing, from the desktop to the data center, in the 1990s and beyond. With Alpha, Digital will provide customers with technology, systems, and services to achieve current and long-term business objectives and gain maximum competitive advantages.

68000 is a trademark of Motorola, Inc.

CORP/92/524d

To Distribution List:

Author: COLETTE CREWS
Date: 13-Feb-1992
Posted-date: 21-Feb-1992
Precedence: 1
Subject: Final Alpha Press Release

Sarah Miller
(508) 264-5420

Barbara Watterson
(508) 568-6501

DIGITAL ANNOUNCES THE ALPHA OPEN COMPUTING ARCHITECTURE, THE WORLD'S FASTEST MICROPROCESSOR, AND NEW BUSINESS PRACTICES

...Lays the foundation for 21st century network computing with a solid bridge from the present.

HUDSON, MA -- February 25, 1992 -- Digital Equipment Corporation today announced Alpha, its program for 21st century computing. According to Kenneth H. Olsen, President of Digital Equipment Corporation, "Alpha is a totally new, open computing architecture that will be the foundation for advanced 21st century computing. It will give computer users a clear and consistent growth path from today's computing technology to the benefits of advanced 21st century computer technology. The Alpha program is a major element in the new, more competitive Digital Equipment Corporation, and we believe it will significantly fuel Digital's growth in the coming years."

"This new architecture will, over time, address the needs of a broad range of computer users by providing systems that span the desktop to the supercomputer. Alpha will offer users the flexibility to deploy current applications on popular

-more-

operating environments, beginning with OSF/1 and VMS. It will enhance and extend the capability of today's Digital products. Customers can continue to buy today's leadership VMS and UNIX systems from Digital knowing they have a clear entry path into 21st century computing," added William Demmer, vice president of Digital's VAX VMS Systems and Servers group. "The beauty of Alpha is that it opens the future with a solid bridge from the present," he added.

Announced today were:

- The Alpha architecture. This advanced, full 64-bit Reduced Instruction Set Computing (RISC) architecture is optimized for speed, engineered to support multiple operating systems, and designed to increase performance by a factor of 1000 over its anticipated 25-year life.

- The first Alpha product -- Digital's 21064-AA RISC microprocessor. This 150-MegaHertz microprocessor is the first in a family of full 64-bit chips with address space many thousands of times larger than 32-bit implementations from IBM, Hewlett-Packard, and Sun. The new chip has demonstrated performance at 200 MegaHertz, and over time Digital will offer versions of this microprocessor at various speeds.

  Evaluation quantities of the 21064-AA microprocessor are available now. It is priced at $3375 each in units of 1 to 100; $1650 in units of 101 to 1,000; and $1559 for over 1,000. Quantity shipments will begin in July, 1992.

  The new RISC microprocessor chips are manufactured at the company's high-volume, state-of-the-art manufacturing plants in Hudson, Massachusetts, and South Queensferry, Scotland.

-more-

- New business practices to achieve the broadest possible use of the Alpha architecture, and to make the widest range of software available on Alpha. Digital will sell Alpha at all levels of integration -- chip, board, and system -- to other computer companies and to OEMs. Digital also will license its operating systems (including DEC OSF/1 and VMS), compilers, and layered software products.

  To ensure a broad applications portfolio, Digital is offering a comprehensive support program to help leading software companies quickly move up to Alpha. Thirty Alpha Upgrade Centers are being located in the US, Europe and Asian Rim, staffed by software support personnel with expertise in VMS, ULTRIX, and DEC OSF/1. Complete documentation, seminars and training session are available now, and seed Alpha system will be available this summer. The Alpha Upgrade Program already is underway with many leading software companies. Major vendors supporting the Alpha platform will be highlighted at DECWORLD '92 beginning in April.

- A variety of ways for customers to enhance their VMS and UNIX computing environments with Alpha. Customers can develop and run applications on today's leadership VAX and DECsystem products with the assurance that they easily will be able to add Alpha systems into their computing environments. Alpha systems will provide data, image, source code, and user interface compatibility with VAX systems running open VMS and DECsystem products running DEC OSF/1. Alpha systems will network with these products to share information and processing tasks, and will work in VAXcluster configurations. Common buses for VAX, DECsystem, and Alpha systems means that many peripherals will be able to be used between systems.

- A complete portfolio of service programs for users and other vendors who incorporate Alpha technology into their products. Digital will support customers with services ranging from consulting to training, systems management, and integration and upgrade services. Digital will support vendors during product design and implementation. Through OEMs, VARs, or direct distribution channels, Digital also will provide services for vendors' Alpha-based products.

-more-

Alpha will extend the ability of Digital's current software systems environment to address major business opportunities. Supercomputer or large mainframe applications -- seismic data analysis, econometric forecasting, molecular modeling, engineering design verification, and many others -- will be able to run on Alpha at a fraction of traditional mainframe price per unit of performance. Alpha will expand the market for networks of high performance distributed production systems and servers, replacing mainframes. The powerful new single-chip Alpha microprocessor will open new markets for embedded OEM applications and board-level products. Alpha will provide the power for emerging personal use applications that are beyond the capability of most of today's PCs, such as voice and video, visualization systems, imaging, and artificial intelligence. With Digital's NAS (Network Application Support), customers will have the flexibility to integrate leading-edge, Alpha-based applications with their current applications environment.

"Alpha is based on the expertise Digital has gained from 35 years of providing systems and networking based on leadership architectural design," noted Robert B.Palmer, Digital's vice president of manufacturing. "In addition, Digital has 15 years of experience in designing and delivering chip technology,

including leading-edge CMOS technology. We have some of the industry's most advanced fabrication technology, and most experienced chip design teams using a leading-edge suite of CAD tools, involved in delivering the world's fastest RISC microprocessor."

Digital Equipment Corporation, headquartered in Maynard, Massachusetts, is the leading worldwide supplier of networked computer systems, software, and services. The company pioneered and leads the industry in interactive, distributed and multivendor computing. Digital and its partners deliver the power to use the best integrated solutions - from desktop to data center - in open information environments.

###

Note to Editors: DEC, DECsystem, DECWORLD, ULTRIX, VAX, VAXcluster and VMS are trademarks of Digital Equipment Corporation.

OSF/1 is a trademark of Open Software Foundation, Inc.

UNIX is a registered trademark of UNIX System Laboratories, Inc.

Hewlett-Packard is a registered trademark of Hewlett-Packard Company.

IBM is a registered trademark of International Business Machines Corporation.

SUN is a registered trademark of Sun Microsystems, Inc.

CORP/92/524a

DEC 3000 Model 400 AXP Workstation

The World's First Desktop Workstation with Over 100 SPECmark Performance

digital

- Industry-leading desktop price/performance based on the Alpha AXP™ architecture
- Full 64-bit architecture breaks through the limitation of 32-bit systems to solve the most complex problems
- The application platform of choice for professional software developers
- Affordable 3D graphics for the desktop
- The ideal workstation to upgrade to from your existing desktop systems

The DEC 3000 Model 400 AXP™ workstation introduces over 104 SPECmark performance to the desktop. For the first time, you now have the power to solve complex problems where 32-bit systems have limited you in the past. 64-bit computing can deliver what your business needs today, and will put emerging technologies, such as multimedia, to work for you in the future.

Optimized for today's application developer, the DEC 3000 Model 400 AXP also sets the pace for data analysis, CAD/CAM, and large volume commercial applications. Additionally, it can accommodate your mainstream 3D graphics requirements.

The DEC 3000 Model 400 AXP workstation provides for expansion of memory, storage, I/O, and graphics. The system provides a 90 Mbyte-per-second I/O bus with three graphics and expansion slots, internal disk storage of over two Gbytes, and up to 512 Mbytes of memory. All of this in a box that fits comfortably on your desk.

digital

Features	Benefits
64-bit computing	Vast amounts of data and complex business problems are easily handled so you can respond quicker to your business needs.
Choice of DEC OSF/1® AXP or OpenVMS AXP operating systems	Choose from a base of thousands of existing applications.
Vast array of software development tools, such as compilers and translators, available from Digital and others	Provides the ideal complete application development environment on your desk.
Fast, low-cost graphics with multiscreen expansion	Provides economical, visual representation of your data.
High-speed TURBOchannel I/O bus	You can connect to the options you need to get your job done.

Alpha AXP Systems Protect Your Investment

If you are using a VAXstation or DECstation, Digital's unparalleled investment protection program provides a cost-effective path when *you're* ready to make the move to Alpha AXP systems.

Outstanding Service from an Industry Leader

To complete your total solution requirements, Digital offers a full range of consultation and services for networking, training, and software support. Digital offers a variety of full one-year product warranties so you can tailor the support according to your unique needs. Whatever the service solution, you benefit from Digital, the single point of contact.

For More Information

To learn more about the DEC 3000 Model 400 AXP workstation or any of Digital's systems and services, contact your local Digital sales representative or business partner.

Digital believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. Digital is not responsible for any errors in the information given in this publication.

The following are trademarks of Digital Equipment Corporation: Alpha AXP, AXP, the AXP logo, the AXP signature, DEC, DECchip, DECnet, DECstation, DECwindows, OpenVMS, the DIGITAL logo, TURBOchannel, and VAXstation.

Prestoserve is a trademark of Legato Systems, Inc. NFS is a registered trademark of Sun Microsystems, Inc. Motif and OSF/1 are registered trademarks of Open Software Foundation, Inc. SPEC and SPECmark89 are trademarks of the Standard Performance Evaluation Corporation.

DEC 3000 Model 400 AXP Workstation Specifications	
SPECmark89™	104+
Dhrystone MIPS	134
LINPACK 100x100 DP MFLOPS	26
CPU	DECchip 21064 RISC Microprocessor
Clock Speed	133 MHz
On-chip Cache	8 Kbytes of instruction/8 Kbytes of data
On-board Cache	512 Kbytes, write back
Maximum I/O Throughput	TURBOchannel: 90 Mbytes/sec
I/O Support Included	Three TURBOchannel Slots, Two SCSI-2 Controllers, Ethernet, ISDN;* Synch/Asynch w/Modem Control, DEC 423 Printer Port, Voice-Quality Audio
I/O Support Optional	FDDI, Prestoserve™ *
Maximum Memory	128 Mbytes (512 Mbytes*)
Maximum Storage	Internal 2.1 Gbytes; Total: 9.5 Gbytes
Software Environment	– DEC OSF/1 AXP operating system; TCP/IP, NFS® – OpenVMS AXP operating system; NAS 250; DECnet – DECwindows Motif®
Graphics	Accelerated 2D, 8/24 Planes High-Performance Accelerated 3D, 8/24 Planes
Operating Environment	
Temperature	10°-40°C (50°-104°F)
Relative Humidity	10%-90% (noncondensing)
Maximum Operating Altitude	2400 m (8000 ft)
Power Requirements	
Line Voltage: 120 V/240 V	Voltage Tolerance: 88-132 V/176-264 V
Frequency – Single Phase: 50 Hz/60 Hz	Frequency Tolerance: 47-63 Hz
Max. Running Current: 8.5A/4.25A	Max. Power Consumption: 420 W
Physical Characteristics	
Desktop System	Height: 12.7 cm (5.0 in); Width: 50.8 cm (20.0 in); Depth: 44.5 cm (17.5 in); Weight: 20.5 kg (45.0 lb)

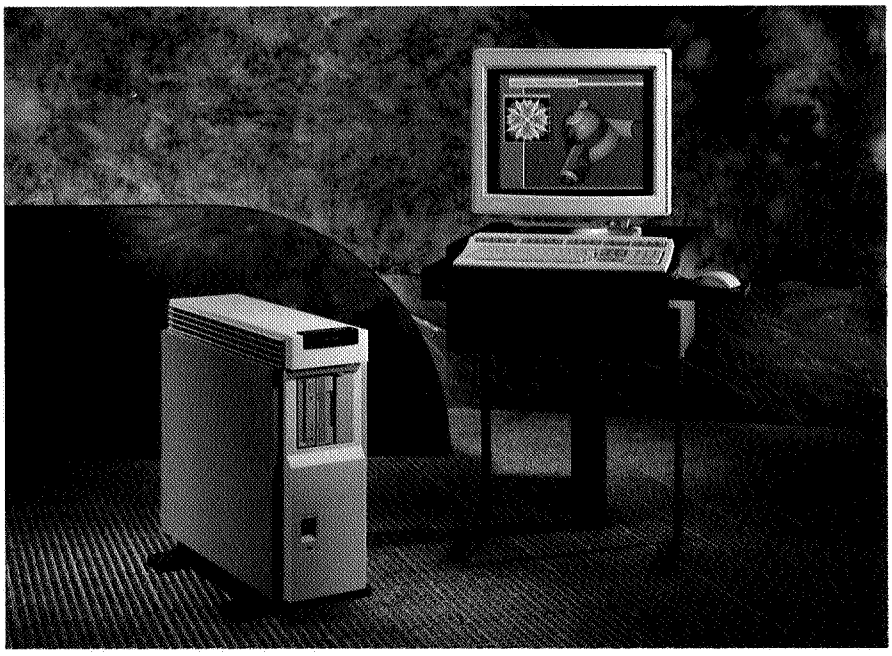
*Available with upcoming operating system release.

Printed in U.S.A. EC-F2078-51 REL# 189/92 11 72 100.0 DAS/MLO Copyright 1992 Digital Equipment Corporation. All Rights Reserved.

DEC 3000 Model 500 AXP Workstation

The World's Most Powerful and Versatile Workstation

digital



- Superior power – Alpha AXP™ starts where other architectures leave off
- Full 64-bit architecture breaks through the limitation of 32-bit systems to solve the most complex problems
- Unmatched expansion capability in memory, storage, networking, and graphics
- The 3D engine of choice for visualization and animation
- Upgrade your current workstation to state-of-the-art technology

The DEC 3000 Model 500 AXP™ workstation is the most powerful workstation in Digital's 64-bit Alpha AXP family. Today's business climate requires you to take advantage of every opportunity. You must analyze vast amounts of data. You can't afford to wait for your computing system to catch up with your business needs. You must respond quicker to the market with accurate decisions and data. What you need is 64-bit computing – to deliver what your business needs today, and to put emerging technologies, such as multimedia, to work for you in the future. 64-bit computing puts solutions to your needs within your reach. With over 118 SPECmark performance, the DEC 3000 Model 500 AXP workstation enables you to gain the competitive advantage you need, whatever your business.

More than just power, the DEC 3000 Model 500 AXP gives you the greatest expansion capability available in the industry today. It provides a 100 Mbyte-per-second I/O bus with six option slots for graphics and expansion, internal disk storage capacity of over four Gbytes, and up to one Gbyte of memory. For custom configurations, it is also available as a rackmountable workstation.

Designed for today's most demanding applications such as computational fluid dynamics, molecular modeling, and simulation of complex systems, the DEC 3000 Model 500 AXP workstation includes an integral 2D graphics accelerator, and can accommodate your most powerful 3D graphics requirements.

digital

Features	Benefits
64-bit computing	Vast amounts of data and complex business problems are easily handled so you can respond quicker to your business needs.
Choice of DEC OSF/1® AXP or OpenVMS AXP operating systems	Choose from a base of thousands of existing applications.
Most expandable memory and storage	You can run your complex applications with no limitations.
Wide choice of high-performance graphics options	Your graphics applications run faster – improving your productivity. You can add power and capabilities as you need them.
Fast CPU combined with fast I/O, storage, graphics, and networking	Timely data and problem solving enables you to enhance your business with improved accuracy and lower costs.

Alpha AXP Systems Protect Your Investment

If you are using a VAXstation or DECstation, Digital's unparalleled investment protection program provides a cost-effective path when *you're* ready to make the move to Alpha AXP systems.

Outstanding Service from an Industry Leader

To complete your total solution requirements, Digital offers a full range of consultation and services for networking, training, and software support. Digital offers a variety of full one-year product warranties so you can tailor the support according to your unique needs. Whatever the service solution, you benefit from Digital, the single point of contact.

For More Information

To learn more about the DEC 3000 Model 500 AXP workstation or any of Digital's systems and services, contact your local Digital sales representative or business partner.

Digital believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. Digital is not responsible for any errors in the information given in this publication.

The following are trademarks of Digital Equipment Corporation: Alpha AXP, AXP, the AXP logo, the AXP signature, DEC, DECchip, DECnet, DECstation, DECwindows, OpenVMS, the DIGITAL logo, TURBOchannel, and VAXstation.

Prestoserve is a trademark of Legato Systems, Inc. Motif and OSF/1 are registered trademarks of Open Software Foundation, Inc. NFS is a registered trademark of Sun Microsystems, Inc. SPEC and SPECmark89 are trademarks of the Standard Performance Evaluation Corporation.

Front page photo: Screen display courtesy of PTC.

DEC 3000 Model 500 AXP Workstation Specifications		
SPECmark89™	118+	
Dhrystone MIPS	151	
LINPACK 100x100 DP MFLOPS	30	
CPU	DECchip 21064 RISC Microprocessor	
Clock Speed	150 MHz	
On-chip Cache	8 Kbytes of instruction/8 Kbytes of data	
On-board Cache	512 Kbytes, write back	
Maximum I/O Throughput	TURBOchannel: 100 Mbytes/sec	
I/O Support Included	Six TURBOchannel Slots, Two SCSI-2 Controllers, Ethernet, ISDN;* Synch/Asynch w/Modem Control, DEC 423 Printer Port, Voice-Quality Audio	
I/O Support Optional	FDDI, Prestoserve™*	
Maximum Memory	256 Mbytes (1 Gbyte*)	
Maximum Storage	Internal: 4.2 Gbytes; Total: 11.6 Gbytes	
Software Environment	– DEC OSF/1 AXP operating system; TCP/IP, NFS® – OpenVMS AXP operating system; NAS 250; DECnet – DECwindows Motif®	
Graphics	Accelerated 2D, 8/24 Planes High-Performance Accelerated 3D, 8/24 Planes	
Operating Environment		
Temperature	10°-40°C (50°-104°F)	
Relative Humidity	10%-90% (noncondensing)	
Maximum Operating Altitude	2400 m (8000 ft)	
Power Requirements		
Line Voltage: 120 V/240 V		Voltage Tolerance: 88-132 V/176-264 V
Frequency – Single Phase: 50 Hz/60 Hz		Frequency Tolerance: 47-63 Hz
Max. Running Current: 10.0A/5.0A		Max. Power Consumption: 686 W
Physical Characteristics		
Pedestal	Height: 62.7 cm (24.7 in); Width: 22.3 cm (8.8 in); Depth: 75.4 cm (29.7 in); Weight: 35.0 kg (77.0 lb)	
Rackmount	Height: 21.3 cm (8.4 in); Width: 44.5 cm (17.5 in); Depth: 68.6 cm (27.0 in); Weight: 29.5 kg (65.0 lb)	

*Available with upcoming operating system release.

Printed in U.S.A. EC-F2075-51 REL# 188/92 11 72 100.0 DAS/MLO Copyright 1992 Digital Equipment Corporation. All Rights Reserved.

Alpha AXP Systems Summary

A Complete Family of 64-bit, RISC, Open Systems

digital

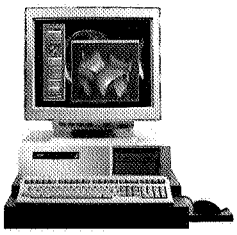
The Alpha AXP™ family is an extensive range of fully compatible systems, from the desktop to the data center. Alpha AXP systems are based on the DECchip 21064, the industry's fastest microprocessor. The innovative performance of Alpha AXP systems allows them to handle your toughest technical, commercial, scientific, and business-critical applications. With their ability to enhance today's applications *and* handle the most demanding future applications, Alpha AXP systems provide solutions that give you a competitive business advantage.

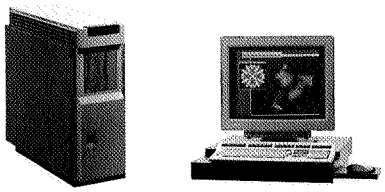
The Alpha AXP family is a universal platform that offers flexibility – the flexibility to make choices. For example, AXP systems support the OpenVMS AXP, DEC OSF/1 AXP, and, soon, Microsoft® Windows™ NT AXP operating systems.

You can take full advantage of the high speed of the Alpha AXP microprocessor because it is combined with advanced, high-speed I/O subsystems, large memory capacities, and the latest storage devices to create very well-balanced systems. The system's performance is complemented by low initial cost and low maintenance cost to create a level of price/performance that leads the industry.

Mission-critical applications keep running around the clock thanks to high availability features and complete data protection. Designed for uptime, Alpha AXP systems are complemented by built-in features that ensure data integrity, security, and reliability. And if your needs grow, they offer you many ways to expand, including symmetric multiprocessing, networking, and clustering.

Alpha AXP System Comparison Chart





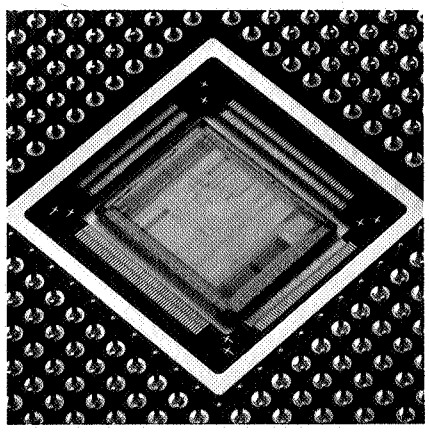
System	DEC 3000 Model 400 AXP Desktop Workstation and Model 400S AXP Desktop System	DEC 3000 Model 500 AXP Deskside Workstation and Model 500S AXP Deskside System
CPU Features		
Number of Processors	1	1
CPU/Clock Speed	DECchip 21064/133 MHz	DECchip 21064/150 MHz
Cache Size (On Chip/On Board)	8 KB I-cache, 8 KB D-cache/512 KB	8 KB I-cache, 8 KB D-cache/512 KB
In-Cabinet CPU Upgrade	N/A	N/A
Performance		
SPECfp92	112.2	126.0
SPECrate_fp92	2631.6	2967.4
SPECint92	65.3	74.3
SPECrate_int92	1549.3	1762.1
SPECmark89/SPECThruput89	108.1	121.5
Dhrystones/sec (V1.1/V2.1)	228,310/249,626	257,731/281,214
LINPACK 100 × 100 (DP MFLOPS)	26.4	30.2
LINPACK 1000 × 1000 (DP MFLOPS)	70.8	79.9
I/O Features		
Maximum Memory Capacity (4 Mbit-chip/16 Mbit-chip)	128 MB/512 MB*	256 MB/1 GB*
Maximum Disk Capacity (in cabinet/total)	2.1 GB/9.5 GB	4.2 GB/11.6 GB
Maximum I/O Throughput	90 MB/s	100 MB/s
I/O Support	Both Systems: 2 SCSI-2, 3-slot TURBOchannel, Ethernet, FDDI, ISDN,* Prestoserve,* DECram Workstation: Voice-quality Audio	Both Systems: 2 SCSI-2, 6-slot TURBOchan Ethernet, FDDI, ISDN,* Prestoserve,* DEC Workstation: Voice-quality Audio
Graphics	HX, TX,* PXG+,* PXGT+*	HX, PXG+,* PXGT+*
High Availability Features		
Cluster Support	Ethernet,* FDDI*	Ethernet,* FDDI*
High Availability Features Supported	Disk Shadowing*	Disk Shadowing*
Software Features		
System Software	OpenVMS AXP, DEC OSF/1 AXP	OpenVMS AXP, DEC OSF/1 AXP

*Available with upcoming operating system release.

**8 SCSI-2 controllers supported initially.

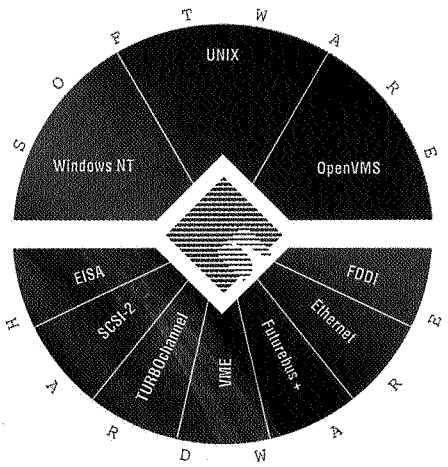
†Performance scales linearly for multiple processors.

The Fastest CPU Chip in the Industry



Alpha AXP systems owe their lightning-fast performance to the DECchip 21064 microprocessor. The chip has been designed to accommodate high-performance features like multiple instruction issue and symmetric multiprocessing. Systems using the DECchip 21064 today range from low-cost desktop systems to powerful mainframe-class systems. A fast chip means Alpha AXP systems provide the best performance across a wide range of industry-standard benchmarks such as the SPEC92 suite of benchmarks, as well as real-world applications.

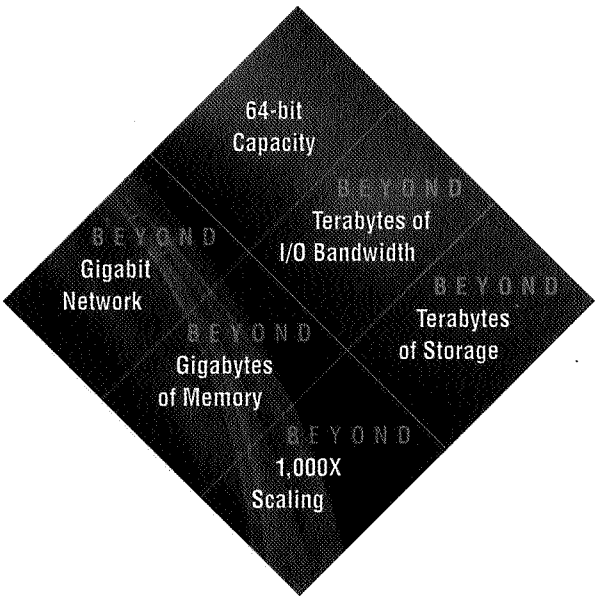
Open Choices With the Universal Platform

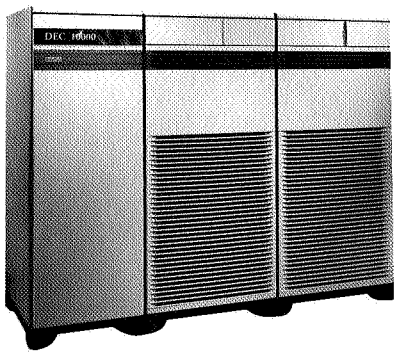


The Alpha AXP universal platform offers a choice of three of the most significant software environments in the industry: Microsoft's Windows NT, UNIX, and OpenVMS. Because no single platform can meet all customer requirements, Digital offers a choice of leadership operating systems, each bringing with it a huge selection of applications from Digital and its business partners. The Alpha AXP universal platform also provides access to a practically limitless choice of hardware peripherals via any of the popular industry-standard interconnects.

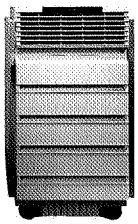
Longevity Built In

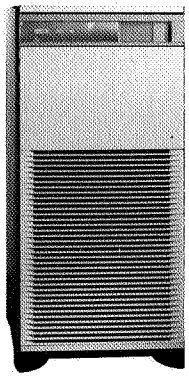
While other vendors' RISC architectures are already many generations old, the new Alpha AXP architecture has been designed to last 25 years. With 64 bits, the Alpha AXP architecture can handle especially demanding applications that need address space that is larger than 32 bits, and that need exceptionally high performance. The performance and capacity go far beyond traditional limits and take you into the 21st century today. The architecture's unprecedented longevity means your computing costs will be lowered for decades to come.





DEC 10000 AXP Mainframe-Class System	System
	CPU Features
Model 610: 1, Model 620: 2, Model 630: 3, Model 640: 4, Model 650: 5, Model 660: 6	Number of Processors
DECchip 21064/200 MHz	CPU/Clock Speed
8 KB I-cache, 8 KB D-cache/4 MB per processor	Cache Size (On Chip/On Board)
Each DEC 10000 AXP system upgrades to any higher DEC 10000 AXP system	In-Cabinet CPU Upgrade
	Performance
Model 610: 200.4	SPECfp92
Model 640: 17,187.2	SPECrate_fp92
Model 610: 106.5	SPECint92
Model 640: 9107.9	SPECrate_int92
Model 610: 184.1, Model 640: 654.6	SPECmark89/SPECThruput89
Model 610: 342,465/377,359†	Dhrystones/sec (V1.1/V2.1)
Model 610: 42.5†	LINPACK 100 × 100 (DP MFLOPS)
Model 610: 111.6†	LINPACK 1000 × 1000 (DP MFLOPS)
	I/O Features
2 GB/14 GB*	Maximum Memory Capacity (4 Mbit-chip/16 Mbit-chip)
56 GB/200 GB (Over 10 TB*)	Maximum Disk Capacity (in cabinet/total)
400 MB/s	Maximum I/O Throughput
4 12-slot XMI, 3 9-slot Futurebus+,* 10 CI,* 24 DSSI,* 24 SCSI-2,** 16 Ethernet, 8 FDDI, SDI, Prestoserve,* HiPPI,* IPI,* VME,* DECram	I/O Support
N/A	Graphics
	High Availability Features
Ethernet,* DSSI,* CI,* FDDI*	Cluster Support
Disk Shadowing,* N+1 Redundant Power System, Integrated Uninterruptible Power System, Integrated Power Conditioning, Disk Warm Swap	High Availability Features Supported
	Software Features
OpenVMS AXP, DEC OSF/1 AXP	System Software





DEC 4000 AXP Distributed/Departmental System	DEC 7000 AXP Data Center System
Model 610: 1; Model 620: 2	Model 610: 1, Model 620: 2, Model 630: 3, Model 640: 4, Model 650: 5, Model 660: 6
DECchip 21064/160 MHz	DECchip 21064/182 MHz
8 KB I-cache, 8 KB D-cache/1 MB per processor	8 KB I-cache, 8 KB D-cache/4 MB per processor
Model 610 upgrades to Model 620	Each DEC 7000 AXP system upgrades to any higher DEC 7000 AXP system
143.1	Model 610: 182.1
Model 610: 3317.0, Model 620: 6214.5	Model 610: 4126.0, Model 620: 8135.1, Model 630: 11,859.8, Model 640: 15,739.4
Model 610: 83.5	Model 610: 96.6
Model 610: 1985.8, Model 620: 3816.1	Model 610: 2188.6, Model 620: 4291.4, Model 630: 6306.5, Model 640: 8366.8
Model 610: 136.2, Model 620: 248.8	Model 610: 167.4, Model 620: 308.7, Model 630: 454.4, Model 640: 604.4
Model 610: 279,329/303,951†	Model 610: 311,526/343,643†
Model 610: 36.3†	Model 610: 38.6†
Model 610: 86.4†	Model 610: 102.1†
512 MB/2 GB*	2 GB/14 GB*
16 GB/56 GB	28 GB/284 GB (Over 10 TB*)
160 MB/s	400 MB/s
4 SCSI-2, Fast SCSI-2,* 4 DSSI, 6-slot Futurebus+, Ethernet, FDDI,* Prestoserve,* HiPPI,* IPI, VME,* DECram	4 12-slot XMI, 3 9-slot Futurebus+,* 10 CI,* 24 DSSI,* 24 SCSI-2,** 16 Ethernet, 8 FDDI, SDI, Prestoserve,* HiPPI,* IPI,* VME,* DECram
N/A	N/A
Ethernet,* DSSI,* FDDI*	Ethernet,* DSSI,* CI,* FDDI*
Disk Shadowing,* Uninterruptible Power Supply, Disk Warm Swap*	Disk Shadowing,* N+1 Redundant Power System, Integrated Uninterruptible Power System, Integrated Power Conditioning, Disk Warm Swap
OpenVMS AXP, DEC OSF/1 AXP	OpenVMS AXP, DEC OSF/1 AXP

Performance is highly dependent on configuration, application, and operating environment. Individual workloads should be carefully evaluated before making performance estimates for specific applications. In this chart, no warranty of system performance is expressed or implied.

Features may differ between OpenVMS AXP and DEC OSF/1 AXP systems.

Details on configurations are found in the *Digital Systems and Options Catalog.*

digital

For More Information

For more information on the Alpha AXP family and Digital's many other products, please contact your local Digital representative or reseller. With sales and service offices located all over the globe, Digital can provide the information you need to become more productive.

Digital believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. Digital is not responsible for any inadvertent errors.

Digital will conduct its business in a manner that conserves the environment and protects the safety and health of its employees, customers, and the community.

The following are trademarks of Digital Equipment Corporation: AXP, Alpha AXP, the AXP logo, the AXP signature, DEC, DECchip, DECwindows, the DIGITAL logo, DSSI, OpenVMS, TURBOchannel, VAX, VAXBI, VAXcluster, VMS.

Third-party trademarks: Windows is a trademark and Microsoft is a registered trademark of Microsoft Corporation. OSF and OSF/1 are trademarks of the Open Software Foundation. Prestoserve is a registered trademark of Legato Systems, Inc. SPEC and SPECmark89 are registered trademarks of the Standard Performance Evaluation Corporation.

ALPHA AXP: The breakthrough into twenty-first century computing with the world's fastest commercially available microprocessor

digital

Issue 1. Oct 92

Alpha AXP. An entirely new 64-Bit computer architecture for the next century

As the 20th Century draws to a close, more and more computer power is being needed to drive our extremely complex applications. But business needs aren't static. In the next century, businesses will continue to specify new procedures, techniques and tasks which will require even more powerful computers. Where do we go from here?

The ageing proprietary architectures that have sustained our needs so far, are beginning to show their limitations. This raises a whole set of questions which must be answered by the forward looking computer professional: how can we increase performance without making computer power too expensive to be viable? How can we take advantage of new technology without being locked into one supplier? How do we make the transition to new superpowerful computers without disrupting present business procedures or endangering existing software investments?

What is the next step?

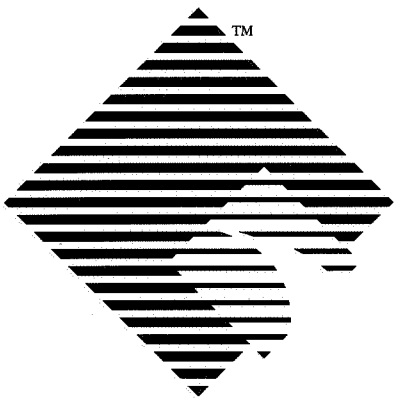
Digital began to consider these questions several years ago. Millions of dollars and thousands of man-hours went into a massive effort to find the answers. And the answer Digital has come up with is Alpha AXP. Alpha AXP is a brand new computer architecture that provides leading-edge performance at a viable expense now and for the foreseeable future. Like Digital's long-lived VAX architecture, Alpha AXP is designed to take advantage of new technologies as and when they become available. In fact the basic Alpha AXP architecture is expected to have a life span of up to twenty-five years. Twenty-five years in which an AXP system will always stay at least one step ahead of the demands of a rapidly changing business world.

But the Alpha AXP programme hasn't turned its back on the past. It has always been Digital's policy to protect our customers' existing software investments, so Alpha AXP offers full binary application compatibility. If an application will run on AXP systems today you can be completely certain that it will run on every successive generation of the architecture that is ever introduced.

The programme's open philosophy means that the architecture not only supports UNIX, Microsoft Windows NT and OpenVMS, but also that Digital will be licensing it to anyone who wants it at whatever level of complexity they desire (from single chips to massively parallel multiprocessing systems).

Understanding how all this is possible is easier if you remember who came up with Alpha AXP: Digital.

Digital is one of the world's major IT suppliers and semiconductor manufacturers and has a huge investment in research, services and consultancy expertise. Our investment in Alpha AXP shows a commitment to the future that is unsurpassed. For instance, even before the first AXP system becomes available, we have already invested massive amounts in a new plant to build the next generation of processors. And it doesn't end there. Each time we produce a new generation, we'll have to do it again.



Alpha AXP

F A C T

Designed for a twenty- five year life span, Alpha AXP performance is expected to increase by a factor of more than a thousand during this period

What about transition to Alpha AXP? Digital has the experience to make moving to Alpha AXP trouble-free. After all, we have done it before. Unlike many other manufacturers, we've always provided upgrade paths from one generation of architectures to the next. Which is why fifteen year old VAX applications will run without modification on the very latest VAX system.

Now you know more about who's behind the next giant step in computing, read on to find out more about the breadth of the Alpha AXP programme, and what it can do for you.

Alpha AXP and the Software Developer

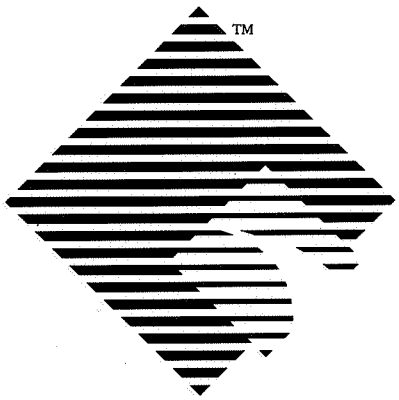
Microprocessors based on Alpha AXP have huge benefits for the software developer. Firstly, Alpha AXP is an enabling technology - it allows you to create software that was previously thought unachievable due to cost, operating system restrictions and limitations on memory and speed.

Secondly, any application whether new or ported to Alpha AXP will run faster. This increase in speed allows you to add more features to existing applications without compromising performance.

Thirdly, Alpha AXP has been designed to embody an open philosophy from the outset. This means that existing applications designed for UNIX, OpenVMS and Windows for DOS will port to Alpha AXP very easily. Applications designed for other operating systems will follow as soon as their vendors port to Alpha AXP. This instantly creates a bigger, broader market for new power-hungry applications. But that's not all. Openness also means that your investments in existing software are fully protected. You won't have to throw good applications away when you port to Alpha AXP.

Since its launch, the architecture has generated huge interest from end-users keen to have products based on Alpha AXP. The major software houses have picked up on this interest and many are already porting to Alpha AXP. The applications they design will have an in-built, commanding lead over competitive applications which don't port to Alpha AXP. Shouldn't your applications be amongst the leaders? After all, Digital has taken steps to make porting as easy as possible.

For instance, there are thirty porting centres world-wide (with more to follow) where Digital's expertise and full range of services is available to help you migrate to Alpha AXP. Education, training, integration and migration services, client/server systems - all these are already in place to help ease your transition to Alpha AXP.



Alpha AXP

F A C T

The DEC 21064 chip is a dual instruction processor containing 1.68 million transistors, with a peak speed of 400 million instructions per second

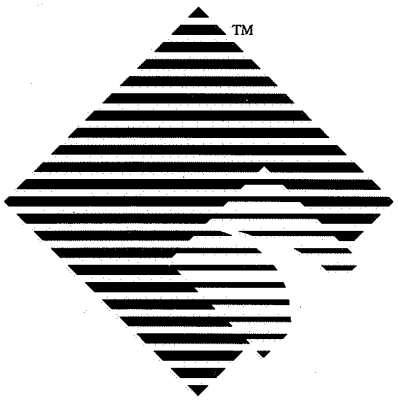
Alpha AXP and the Technical OEM

The first product based on the Alpha AXP architecture is the DEC 21064 RISC microprocessor. Available at all levels of integration (from chip to board to whatever complexity you require), this is the microprocessor that has been attracting attention from leading computer companies like Cray, Kubota, Aeon Systems and Olivetti. They're all licensed to use AXP processors, and there is a long list of those evaluating the processor with a view to joining the bandwagon.

The Alpha AXP architecture's benefits to the technical OEM are many and impressive. Because it has no operating systems bias, AXP products will run UNIX, OpenVMS and Microsoft Windows NT. And there's the potential for others to port many different operating systems to Alpha AXP too (for instance, Real Time Operating System kernels). But that's not all. The architecture has such a lead over existing technology that it will make new user interfaces such as voice and gesture control commercially viable at last.

It's not just operating systems that will benefit from a new openness. Alpha AXP has no programming language bias. If you're used to PASCAL, COBOL, FORTRAN, C, ADA or any of the other leading languages, you won't have to learn a new language to use the Alpha AXP advantage.

But Alpha AXP is not a hot new development from a little known company. Because it comes from Digital Equipment Company, you also have the benefit of a first-class, world-wide support network and the reassurance and reliability of supply from a world-class manufacturer: What more could you ask for?



Alpha AXP

F A C T

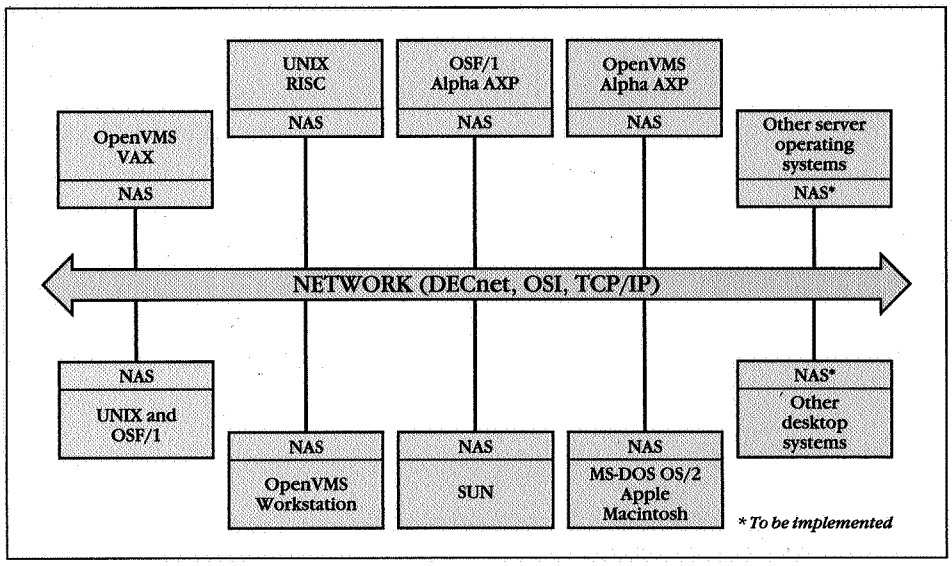
Enables software engineers and designers to exploit fully the most powerful commercially available microprocessor produced this century

Alpha AXP and Openness.
The most open architecture you can buy?

Alpha AXP architecture is designed to permit applications which use standard hardware interfaces to be portable across different systems and platforms. Software applications running on AXP systems will even be able to interact with proprietary systems in a multi-vendor network. How?

Alpha AXP is supported by Network Application Support (NAS), Digital's open-systems building software. NAS supports an extensive range of operating systems from UNIX and OpenVMS to MS-DOS, Mac, OS/2 and SunOS.

This openness, coupled with the use of standard interfaces means that applications are now insulated from the underlying technological differences like different hardware platforms, various operating system biases and restrictions, database limitations and networking protocols.



Digital's implementation of Open Systems standards in NAS allows users to access data and applications on any platform, regardless of system vendor. It protects existing investments, allows freedom of choice of vendor, and keeps an organisation's IT options open for the future.

Open standards for Alpha AXP include:

Languages:	C, COBOL, FORTRAN, PASCAL, ADA.
Platform:	C2, Security, POSIX 1003.1, X/Open XPG-3.
User Interfaces:	GKS3D, OSF/Motif, PHIGS, X Window System.
Documents:	ODA/ODIF.
Data Portability:	SQL.
Networking Standards:	Ethernet, OSI, RPC, TCP/IP, FDDI.

Digital has already evolved a planned partnering policy with software originators to allow powerful new applications to be developed using a wide range of operating environments.

As part of our commitment to openness, operating systems, compilers and layered software products will all be available as licensed products.

Alpha AXP
F A C T

Designed to use standard interfaces, to be portable across many different platforms and to inter-operate with other applications across a multi-vendor network

Openness and Other Vendors

The fact that all levels of Alpha AXP architecture are available for licensing by other vendors is already paying dividends. It has been chosen by Cray Research, the world's foremost supercomputer manufacturer, for its first generation Massively Parallel Processing system (MPP). Kubota Inc. has decided to use Alpha AXP for its high performance graphics workstation. Alpha AXP is currently being used by Aeon Systems for a new generation of board level products for real time applications. And Olivetti is the latest company to license Alpha AXP technology. (Other major hardware manufacturers are currently in the process of evaluating Alpha AXP).

Alpha AXP and In-house Software Developers

The Alpha AXP programme places the power of the world's first 64-Bit chip at the full disposal of the software designer. The awesome power and speed of Alpha AXP opens new horizons, permitting improvement to existing software and new features which were not previously possible due to limitations of speed and memory.

Because Alpha AXP is a scalable family, machine code development time is greatly reduced and the same code will meet an extensive range of applications. Alpha AXP is designed to allow any software application that uses standard interfaces to be portable across many different platforms and to inter-operate with other applications across multi-vendor networks.

With Alpha AXP, Digital has done something unique among chip manufacturers by including compilers as an integral design element of the architecture.

Software vendors already committed to Alpha AXP

a/Soft Development, Inc.
A/S/T/R
A. S. THOMAS, Inc.
AAMPHORA Systems
ABACUS Software, Incorporated
ABB Kraftwerksleittechnik GmbH
ABB Simcon Inc.
ABB Stromberg Power Cy
ABB SYSTEMS CONTROL CO. INC.
ABC Smart Software
ABS - American Business Systems
Accelr8 Technology Corporation
ACCESS Corporation
Access International
ACT Sigmex Ltd.
Acucobol, Inc.
ADAC LABORATORIES
Adaptive Research Corporation
ADINA R & D, Inc.
ADL DATA SYSTEMS, INC.
Admin Tec a.s.
ADMINS, Inc.
ADNET Information Systems
ADRA Systems, Inc.
Advanced Data Management
Advanced Digital Data, Inc.
Advanced System Management, Inc.
Advanced Systems Concepts, Inc.
Advanced Technology Center
Advent Online Knowledge, Inc.
Aeon Systems, Inc.
Aftec, Inc.
AGE Logic, Inc.
AGOSOFT Inc.
AICorp, Inc.
AimTech Corporation
AIS Corp.
AIS Ges.m.b.H
Alfalfa Software, Inc.
Amarex Technology, Inc.
AMDOCS Inc.
American Phoenix Computer Services Inc.
American Turnkey
AMOCAMS/MODULAR
AMS
Anacad, Inc.
Analytical Graphics, Inc.
Anderson Consulting
Ansoft Corporation
Antares Development Corporation
ANTRIM CORPORATION
Apogee Computer Systems, Inc.
Application Programming Techniques Ltd.
Application Systems Corp.
Applied Automation, Inc.
Applied Computer Solutions Inc.
Applied Information Systems
Applied Logic Systems, Inc.
Applied Terravision Systems Inc.
APPX Software, Inc.
ARCADD, Inc.
Array Systems Corp.
Artifical Intelligence Technologies, Inc.
ASA International Ltd.
ASA LEGAL SYSTEMS COMPANY INC.
Ascent Technology, Inc.
Ashland Computer Technology, Inc.
ASK Computer Systems
Aspen Technology, Inc.
ATRION CORPORATION
Austrian Research Centre
AUTOFILE
Automated Financial Technology, Inc.
Automated Office Systems, Inc.
Automated Technology Associates, Inc.
Auto-Scan Systems, Inc.
Aviation Computing Services, Inc.
Axis Computer Systems, Inc.
BAeCAM
BAIRD PETROPHYSICAL INT'L
Basmark Corp.
Battle Green Software, Inc.
BBN Software Products
BDS Systems, Inc.
Beacon Expert Systems, Inc.
Beckman Instruments, Inc.
Bellwether Software Corporation
Bernstein & Associates
BGS Systems, Inc
Biles & Associates
Birmingham Computer Group, Inc.
Bizwaro Corporation
Blossom/Catalytix Corporation
Bormuth Associates
Boston Business Computing, Ltd.
Brain Tree
BRANDT, INC.
Breuer & Co.
Bristol Technology, Inc.
Brock Control Systems, Inc.
BRS Software Products
BSR - Business Systems Resources, Inc.
BusinessWise, Inc.
Buzzwords International Inc.
BV Technologies, Inc.

continued overleaf

C-Pak Corporation
CACI LTD.
CACI Products Company
CADCentre Ltd.
CADENCE DESIGNS SYSTEMS INC.
Cadre Technologies, Inc.
Caine, Farber & Gordon, Inc.
Caldwell-Spartin, Inc.
Calidus Systems Limited
CAM Software, Inc.
CAMBRIDGE ONLINE SYSTEMS LIMITED
CAMEO SOFTWARE SOLUTIONS, INC.
Campus America
Canberra Industries, Inc., Nuclear Products Group
CAP Systems, Inc.
CashHandler Retail Systems Inc.
CEDARDATA PLC
CEGELEC ENTERPRISES
Centera Information Systems, Inc.
Central Area Data Processing
Century Dynamics, Inc.
Cerner Corporation
cfX Incorporated
CHAMPS Software, Inc.
Chemical Design Inc.
Chesapeake Decision Sciences, Inc.
Choice Software Systems Ltd.
Cimage Corporation
CIMCORP Factory Controls Inc.
Cimflex TeKnowedge Corporation
CIMSLMC Incorporated
CIMPLEX Corporation
Cimtek America
Cincom Systems, Inc.
Circuit Rider Connection
CIS
Citymax Integrated Information Systems Ltd.
CLM/Systems, Inc.
-CMI-Competitive Solutions, Inc.
CMSI (Credit Management Solutions, Inc.)
CODA Incorporated
Codar Technology Inc.
COGENT INFORMATION SYSTEMS, INC.
CogniSeis Development, Inc.
Cognition Corporation
Cognition Technology Corp.
Cognos Incorporated
Coherent Systems, Inc.
Collier-Jackson, Inc.
Command Data, Inc.
Commercial Computer Services
Communication Software, Inc.
Compact Software, Inc.
COMPAGNIE GENERALE INFORMATIQUE
CompInfo, Inc.
Compu-Share Business Management Systems
Compunix, Inc.
CompuServe Data Technologies
Computer Aided Decisions, Inc.
Computer Associates International, Inc.
COMPUTER AUTOMATION INC.
COMPUTER ENGINEERING ASSOCIATES, INC.
Computer Generation, Inc.
Computer Management Center
Computer Software Packages
COMPUTERVISION CORPORATION
Computing Information Services, Inc.
COMPUTRON TECHNOLOGIES CORPORATION
COMSHARE,Inc.
Comstow Information Services
Concentration, Heat and Momentum (CHAM)
Consilium, Inc.
Control Systems International
Core Software Technology
Corstar Business Computing Co., Inc.
Cortex Corporation
CPlex Optimization, Inc.
Cray Research, Inc.
CRC International Business Solutions
CSAR Corporation
CSSC, Incorporated
CyberResources Corporation
Cyberscience Corp.
CyberTools, Inc.
Cyborg Systems, Inc.
Cygnet Publishing Technologies, Inc.
Daniel Integrated Software Corporation
Das Consulting, Inc.
Data Blocks
Data Center Software, Inc.
Data Concepts, Inc.
Data Exchange, Inc.
Data Intelligence Systems Corp.
Data Parallel Systems, Inc.
Data Pro Accounting Software, Inc.
Data Research Associates, Inc.
Data Sciences, Inc.
Data Systems & Management
Data Systems, Inc.
Data Systems Support
Data Technical Research, Inc.
Database Publishing Software, Inc.
DATABASE SYSTEMS CORP.
DataBreeze, Inc.
DATABYTE
DataCode Incorporate
Datalogics Incorporated
Datalogix International, Inc.
Datametrics Systems Cor
DataNational Corporation Inc.
Datani a/s
Datatel, Inc.
Data Vantage
Day Data Systems
DAYLIGHT Chemical Information Systems, Inc.
DCS
DDC-I, Inc.
Decathlon Data Systems, Inc.
Decision Analytics
DECISION SYSTEMS, INC.
DEFU data A.m.b.A.
Dekker, Ltd.
DEMAX Software, Inc.
Descartes-RAM
Desktop Data, Inc.
Dialogue, Inc.
Diamond Control Systems
Digital Information Systems Corp.
Digital Insurance Systems Corporation
Digital Tools, Inc.
Dimension Software Systems, Inc.
DIMENSIONAL INSIGHT, INC.
Dimeric Development Corp.
DISC
Discrete Time Systems Corporation
Distribution Architects International
Distribution Management Systems, Inc.
DocuGraphix, Inc.
Document Storage Systems, Inc.
DORN Technology Group, Inc.
Doyle, Munroe Consultants, Inc.
Draves & Barke Systems, Inc.
DSA Systems, DLSA, Inc.
DSD Corporation
DSP Development Corporation
DTC
Dun & Bradstreet Software Services, Inc.
Dynamic Graphics, Inc.
Dynamic Information Systems Corporation
Dynamic Matrix Control Corp.
DYNETICS, Inc.
EA Systems, Inc.
Econintel Treasury Systems, Inc.
EDS/GDS Solutions
EDS/UNIGRAPHICS Division
EEC Systems, Inc.
EFFECTIVE MANAGEMENT SYSTEMS, INC.
EGRET TECHNOLOGIES, INC.
Electrical Engineering Software Inc.
Electronic Cottage Associates
Elite Systems & Peripherals
Ellery Systems, Inc.
Elsid Software Systems, Ltd.
EMERGING TECHNOLOGY CONSULTANTS, INC.
EMPHASYS CORPORATION PTY LTD
Empress Software, Inc.
EMRC TM
Engineering DataXpress, Inc.
Enterprise Technology System
ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE
ENVISION UTILITY SOFTWARE Corp.
Envoy Systems Corporation
Epic Systems Corporation
ERDAS, Inc.
Ergodic Systems, Inc.
ESCA Corporation
ESRI Computing
Escom, Inc.
Evans & Ricker, Inc.
Excalibur Technologies Corporation
Executive Software
EXPERTECH INC.
Expograph bv
Facilities Management Ap
Fauske & Associates, Inc.
FCMC plc
FCX Software Systems
FEA Limited
Federal Software
FEITH Systems and Software, Inc.
Ficke & Associates, Inc.
Financial Accounting Systems,Inc.
Financial Models Company
FINANCIAL SOLUTIONS LIMITED
First Data Corporation, Health Systems Group
Fleming Systems Corporation
Flexus International Corporation
FLOWMASTER INTERNATIONAL LTD
FLUID DYNAMICS INTERNATIONAL
Focus Graphics
Fore Systems Inc.
Foxware Corporation
FRAMASOFT
Fraser Williams Logistics Ltd.
FRONTLINE DISTRIBUTION LIMITED
Funds Associates LTD
FutureSoft, Inc.
Gain Technology, Inc.
GAMMA GROUP INC.
GAMS Development Corp.
Gaylord Information Systems
GC SERVICES, L.P.
GE Fanuc Automation N.A., Inc.
GE Medical/PET Engineering
GEAC COMPUTERS INC.
GENASYS II, INC.
Genetics Computer Group Inc.
Genroco, Inc
Gensym Corporation
Gerber Alley
Gerber Alley Physician Systems
GESCAN International, Inc.
GFI - Informatique, An EDS Company
Gimpel Software
GIST, INC.
Global Health Systems, Inc.
GP Solutions, Inc.
GRAFFMAN ASSOCIATES INCORPORATED
GrayMatter Software Corporation
GRESHAM TELECOMPUTING
Greystone Technology Corporation
GSI
GSI UCOMS
GSI UK MOTORTRADE
GTECH CORPORATION
GWA Information Systems, Inc.
Hancock Software, Inc.
HARLEQUIN
Harnischfeger Engineers, Inc.
Harper & Shuman, Inc.
Harris Electronic Design Automation
HAS AUTOMATION SYSTEMS BV
HAVERLY SYSTEMS, INC.
HBO & Company
Health Care Systems Inc. (HCS)
Health Systems Integration, Inc.
Heuristics Inc.
Hibbitt, Karlsson & Sorensen, Inc.
Hilco Technologies, Inc.
HLP, Inc.
Hogg Robinson Systems
Holistic Systems, Inc.
Humanic Design Corp.
HYPERSOFT EUROPE LIMITED
i-Logix
IBES Corporation
IBUKI
ICAM Technologies Corporation
ICARUS Corporation
ICI
ICONIX Software Engineering, Inc.
IdentiTech, Inc.
IDSI
IDX Systems Corporation
Image Data Corporation
Image Integration Corporation
Imagine Multimedia, Inc.
IMPEX SYSTEMS
IMSL, Inc.
INCOTEL, INC.
Independence Technologies, Inc.
Infisy Systems, Inc.
InfoQuest Systems, Inc.
Information Advantage, Inc.
Information Associates
Information Builders, Inc.
Information Dimensions, Inc.
Information Management Consultants, Inc.
Information Resources, Inc.
Information Spectrum, Inc.
Information Systems Group, Inc.
Information Technology Systems
Informix Software
INGRES
Innosoft International, Inc.
Innovative Interfaces, Inc
Innovative Software
Integrated Distribution Systems,Inc.
Integrated Silicon Systems, Inc. (ISS)
Integrated Software Solutions, Inc.
Integrated Solutions Incorporated
INTEGRATED SYSTEMS, INC
Intelligent Light, Inc.
Intelligent Networks, Inc.
Intellution, Inc.
Ineractive Development Environments,Inc.
Interactive Software Engineering, Inc.
Interactive Software Systems, Inc.
InterConnections, Inc.
Interlake Software Solutions
INTERLEAF Incorporated
INTERNATIONAL DATA SYSTEMS
International Financial Systems Ltd. (IFSL)
International Telesystems Corporation
InterPlant Consulting Inc.
InterSystems
Intex Solutions, Inc.
INTRACO, Inc.
IntraNet, Inc.
INTRIX Systems Group
IPC Technologies, Inc.
IQ Software Corporation
Irvine Compiler Corporation
I.S.E. Inc.
ISG Technologies Inc.
ISYKON Software GmbH
Isys plc
Itasca Systems, Inc.
Ithaca Software
ITS Associates, Inc.
ITS - Process and Software Engineering GES m.b.H.
J. Glaser & Company Incorporated
J.H. Leskin Associates, Inc.
JCI Software
Jefferson Pilot Data Services, Inc.
Jefferson-Pilot Data Services Pty. Ltd.
Jenark Business Systems
Johnson Yokogawa Corporation
JWP Controls
Jyacc, Inc.
Karmak, Inc.
Keane, Inc.
Keystone Information Systems, Inc.
Keywood Office Technologies Ltd.
Knowledge Information Systems
Knowledge Software
KOM Inc.
KPY Corporation
KSH SYSTEMS, INC.
Kubota Pacific Computer, Inc.
LABTECH
Landis & Gyr Powers, Inc.
Large Scale Biology Corporation
Laser Recording Sys, Inc.
Laser-Scan, Incorporated
LBMS, Inc.
Legacy Technology Inc.
Legal IMS, Inc.
Legent Corporation
Light Valve Technology - A Division of Eastman Technology, Inc.
Lindhard Computer Systems A/S
Linian Systems Inc.
LIOCS Corporation
Litton Security Systems
LJK Software
LOGICA
Logical Technology, Inc.
Los Altos Technologies
Lucid, Inc.
Lynggard Pedersen Data A/S
M.A.I.A., Inc
MANCOS COMPUTERS LTD.
Manufacturing and Consulting Systems, Inc.
MARC Analysis Research
Mark V Systems
Maspar Computer Corp.
MAT-MAN SYSTEMS PTY LTD.
Materials Business Systems, Inc. (MBSI)
Mathsoft, Inc.
Matra-Datavision
McCabe & Associates, Inc.
McCue Systems, Inc.
McDonnell Douglas Information Systems Internatonal-Financial Systems
McHugh Freeman
McKEOWN SOFTWARE
MCS, Inc.
MCSS, Inc.

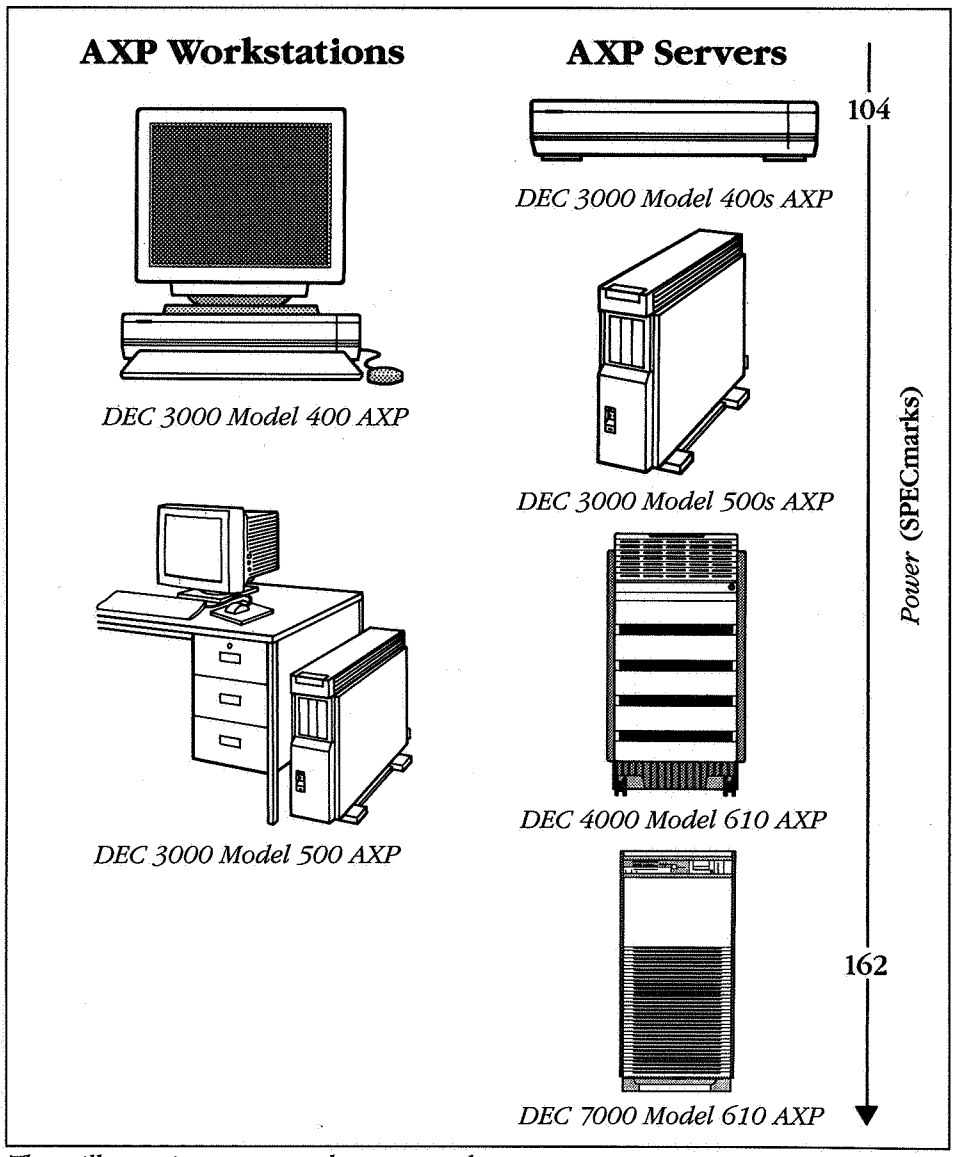
Measurex Management Systems Divison
Medical Information Technology (Meditech)
MEDICAL TECHNOLOGY, INC.
Megabyte Limited
MegaSys Computer Technologies Ltd.
Megaware, Inc.
MEI Software Systems, Inc.
MENTOR GRAPHICS CORP.
Mercury Interactive Corporation
Meta Pharmacy Systems, Inc.
Metrscope Corporation
MFG Systems
MicroCODE Incorporated
MicroImages, Inc.
Microsoft Corporation
Microsystems Engineering Corporation
Midwest Stock Exchange
MIL 3, Inc.
MINCOM Pty Ltd.
Mitech Computer Systems Inc.
Mitech Corporation
M.I.T.I.
MJ Systems, Inc.
MKS, Inc.
MODACAD, INC.
MOLECULAR SIMULATION, INC.
Montagar Software Concepts
MONTRAN
Moore Products Company
MORRISON KNUDSEN CORPORATION
Mtel Technologies
MTS Systems Corporation
Muller Media Conversions, Inc.
MULTI-TEK SOFTWARE CORPORATION
Multihouse Automatisering
Multiware, Inc.
Muscato Corporation
MUST Software International
Nashbar/Associates, Inc.
NATIONAL COMPUTER SYSTEMS
National Data Conversion Institute - NDS
National Information Systems, Inc.
National Investor Data Services, Inc.
Netron, Inc.
Network Computing Corporation
Networking Dynamics Corporation
NEURON DATA
New Tech Information Systems, Inc.
NobelNet, Inc.
Norbete Enterprises, Inc.
North Dakota State University
Northeast Data Systems, Inc.
Northlake Software,Inc.
Northwest Digital Software Inc.
Novacad, Inc.
NOVUS
NPRI,Inc.
NSS Networking Software Specialists
Numerical Alogorithms Group, Inc
Numetrix
OASIS Technology Inc.
Objective Interface Systems, Inc.
Objective Solutions, Inc.
Oil Systems, Inc.
OM Systems International
Omni Industry, A Division of Global Turnkey
OMR Systems Corporation
Omtool, Ltd
OpenAge Ltd
Optimal CAE, Inc.
Oracle Corporation
Order Processing Technologies, Inc.
ORGANIZATION ZOLLOER PAZ
Orion Systems, Inc.
OUROUMOFF INFORMATIQUE SYNFORM
Palette Systems, Inc.
PanData
Panttaja Consulting Group, Inc.
PARALOG INC
Parametric Technology Corporation
Parity Systems Inc.
Pattern Recognition Systems
PAWS, Inc.
PCI Remote Sensing Corp.
PCS Systems, Inc.
PDA Engineering
PE Nelson ,A Division of the Perkin-Elmer Corporation
Pennington Systems Incorporation
PENTA Software Company
Pentanation Enterprises, Inc.
PeopleSoft
Perceptics Corporation
PERFORMANCE SOFTWARE,INC.
Performance Technologies, Inc.
PERFORMIX, Inc.
Pericom Inc.
Phase Three Logic, Inc.
Philip Lieberman & Associates, Inc.
Phoenix Systems
PICKTEL
Piedmont Software, Inc.
Pixar
PRC Public Management Services, Inc.
Precision Visuals, Inc.
Pride Retail Systems, Inc.
Prime Associates, Inc.
PRITSKER COPORATION FACTOR
PRO SYSTEMS, INC.
Pro-Am Software
Process Control Industries
Process Control Systems, Inc.
Process Software, Inc.
Productivity Solutions, Inc.
Proffesional Data Procesing, Inc.
PROFIDEX CORPORATION
PROGRAMIT
Progress Software Corporation
Progress Computer Systems, Inc.
Promis Systems Corporation
PROSIG Informatique Inc.
Prosig USA, Inc.
PS Automatisering BV
PSP Information Group,Inc.
Public Systems Associates, Incorporated
QCPE
QEI Incorporated
Quality Systems Inc.
Quantitive Technology Corporation
Questor Systems, Inc.
Quintus Corporation
Quodata Corporation
R & D Systems
Racal-Redac, Inc.
Radionics Software Applications, Inc.
RADLEY CORPORATION
Raytheon Company
Re:Member Data Services, Inc.
Real-Share Inc.
Real-Time USA, Inc.
Reamdata Incorporated
Recital Corporation
REL-TEK Systems & Design, Inc.
Relational Semantics, Inc.
Research & Planning,Inc.(Research Systems, Inc.)
Resource Systems Corporation
Revolutionary Software
RGTI Systems/Software
Richter Management Services, Inc.
RJN Environmental Associates, Inc.
RMS Technologies, Inc.
Rocket Science, Inc.
Rosetta Technologies, Inc.
Ross Systems, Inc.
RSA, Inc
RSK Consulting, Inc.
RSS Marketing, Inc.
Rugged Digital Systems Inc.
Russell Information Sciences, Inc.
Sabre Systems and Service
SACDA Inc.
Saiga Systems
Salerno Manafacturing Systems, Inc.
Sanchez Computer Assciates, Inc.
Sandata, Inc.
Sandwell Inc., DATAP Systems Division
SAP (UK) LIMITED
SAS Institute, Inc.
SAXPO Incorporated
Scanditronix Inc.
Scandura Intelligent Systems
SCANTECH Corporation
Scheduling Technology Corporation
SCJ Consulting. Inc.
Schlumberger CAD/CAM
Scicon Energy, Inc.
Science Applications International Corporation
SDK Healthcare Information Systems
SDRC Structual Dynamics Research Corporation
Search Software America
Sebern Engineering Inc.
SecaGraphics, Inc.
seiko Instruments, Inc.
SEMA GROUP
Setpoint Inc.
SFERCA
Shakopee Systems, Inc.
SharpImage Software
S.I. Inc.
SI Systems, Inc.
Sidoci enr.
Sigma Design, Inc.
SILMA Incorporatied
SILVACO International
Simpact Associates Inc.
Simucad, Inc.
Simulation Sciences, Inc.
Simutest, Inc.
SIR
SIS Datenverarbeltung Ges.m.b.H
SL Corporation
Smallworld Systems (UK) Ltd.
SmartSystems (UK) Ltd.
SMS
Snow Development Group
Soft Systems Engineering Inc.
SofTrek
SOFTWARE AG
Software Development Systems Inc.
Software House, Inc.
Software Maintenance and Development Systems
Software Moguls, Inc.
SOFTWARE PARTNERS/32, INC.
Software Plus, Inc.
Software Quality Automation
Software Research, Inc.
Solar Computers
Sonnet Software, Inc.
Soren T Lyngso A/S
Source One Solutions Ltd.
SPAR Associates, Inc.
Speakeasy Computing Corporation
Spirit Computer Solutions
Spokane Computer
Springfield Computer Consultants
SPS
SPSS, Inc.
Spyglass, Inc.
SQL Solutions
SQL*Builder Software Company
Square D Company/CRISP Automation Systems
S.R.B. International Ltd.
Stauffer Information Systems
StereoGraphics Corporatopm
STERIA
Stone Mountain Computing
Strategic Alternatives, Ins.
Strategic Systems International
Structured Software Solutions, Inc.
STSC, Inc.
Sullivan System Corporation
Summitpointe Technologies, Inc.
SUNQUEST INFORMATION SYSTEMS
Sunrise Software International
SUNSET SOFTWARE TECHNOLOGY
Swanson Analysis Systems, Inc.
Sybase, Inc.
SYMBIOTIC SOFTWARE
SYMBOLICS, Inc.
SYNERGEN ASSOCIATES, INC.
Synercom Technology, Inc.
SYNOPSYS LIMITED
Synthesis Computer Technologies, Inc.
Systech, Inc.
System Administration & Management, Inc.
System Analysis Corporation
SYSTEM DEVELOPMENT, INC.
System Industries, Inc.
System Insights
SYSTEMA GesmbH & Co KG
Systemetrics, Inc.
Systems & Computer Technology Corporation
SYSTEMS APPROACH INC.
SYSTEMS DESIGNERS SOFTWARE
Systems Innovation Inc.
Systems Strategies, Inc.
T & B Computing
Tactics International, Limited
Talarian Corporation
Target Information Ststems, Inc.
Target Systems Corporation
Tascom, Inc.
Tasking BV
TATE INTEGRATED SYSTEMS, LP
TechGnosis, Inc.
Telco Research Corporation
TELE CONTROL Kommunikations und computersysteme GesmbH
Teledyne Geotech
TeleSoft
TELLUS, Inc.
Templar Technologies
Teradyne, Inc.
Tetra Limited
Texas Instruments, Inc.
Texas Instruments - Process Automation Division
TextBase Imaging, Corp.
TGV, Inc.
The Boston Software Works, Inc.
The CEDRA Corporation
THE DATASTORE INCORPORATED
The Erin Group Incorporated
The MacNeal-Schwendler Corporation
The MathWorks, Inc.
The Parkside Organization, Inc.
The PARSEC Group
The Partners Group, Ltd.
The Sombers Group, Inc.
The Wollongong Group
The Wyndgate Group Ltd.
Thoroughbred Software International
Timeline, Inc.
TMT Group
TOUCH TECHNOLOGIES, INC.
TRADE SERVICE SYSTEMS
Transportable Software Intl., Inc.
Triangle Ernst & Young Inc.
TRIDENT SYSTEMS INC.
TRIFOX, INC.,
Trinary System, Inc.
TSS Systems Corporation
TURN-KEY DISTRIBUTION SYSTEMS, INC.
UCSF
UIS
Unibased Systems Architecture
Unidata, Inc.
UNIFACE
UNIPLEX LTD.
UniPress Software, Inc.
UNIRAS A/S
United Data Systems, Inc.
Unitronix Corp.
University of Georgia
US DESIGN
Users Incorporated
U.S. West Public Safety Group
V-Ststems, Inc.
Valmet Automation, SAGE Division
VARNET The Solution People
VECTOR AND SCALAR PRODUCTS LIMITED
VeraSoft
Verity, Inc.
VG DATA SYSTEMS/FISONS
ViewLogic Ststems, Inc.
Viking Software Services, Inc.
Village Systems
Virtech Inc.
Virtual Software Factory Limited
VISX SOFTWARE INC.
Vista USA, Inc.
VISUAL CYBERNETICS,CORP
Visual Engineering, Inc.
Visual Solutions, Inc.
VLSystems
VOEST-ALPINE Industrieanlagenbau GmbH
W. Quinn Associates
Waterloo Maple Software
Wavetek Corp.
Weir Systems Ltd.
Wilco Communications Inc.
Wilke/Thornton, Inc.
Willow Information Systems
WILSON COMPUTER SYSTEMS
Wingra Technolgies, Inc.
WINTER PARTNERS INC.
WordMARC International Corp.
Xerox Computer Services
Xybion Medical Systems Corporation
XYVision, Inc.
Z/Max Computer Solutions, Inc.
Zia Corporation
ZIFF INFORMATION SERVICES, INC.
ZONIC
Zontec, Inc.
Zortec, Inc.
ZPC Int'l Inc.

This list is growing daily so for the latest list see the Alpha AXP application source book

The First AXP Computer Family

The first systems based on Digital's Alpha AXP architecture will form a family of workstations and servers with PCs and mainframe systems to follow. All the members of the family will share the same commitment to providing optimum performance.

But it's not just microprocessor performance that has been enhanced. By enhancing memory band width, cache size and all aspects of I/O, AXP systems will fully exploit the microprocessors' potential. Networking capabilities have also been increased to ensure overall balanced systems performance.

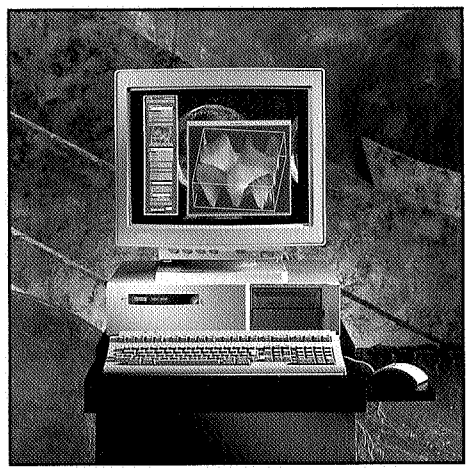


These illustrations are not shown to scale.

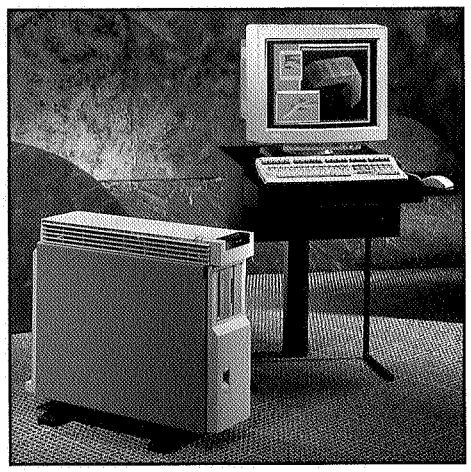
Alpha AXP

F A C T

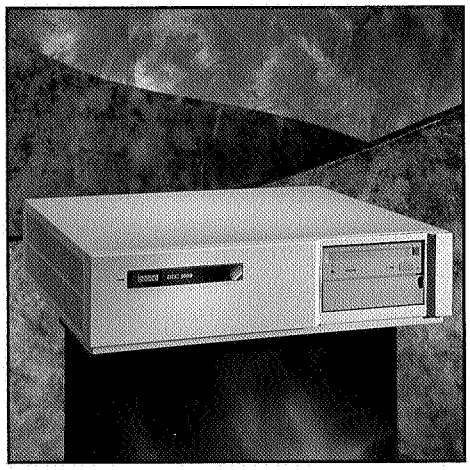
Supports Microsoft Windows NT, UNIX and OpenVMS



DEC 3000 Model 400 AXP
Desktop Workstation

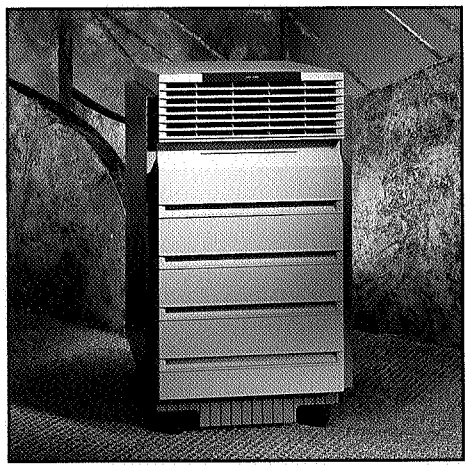


DEC 3000 Model 500 AXP
Deskside Workstation



DEC 3000 Model 400s AXP
Desktop Server

DEC 3000 Model 500s AXP
Deskside Server



DEC 4000 Model 610 AXP
Office Server

DEC 7000 Model 610 AXP
Datacentre Server

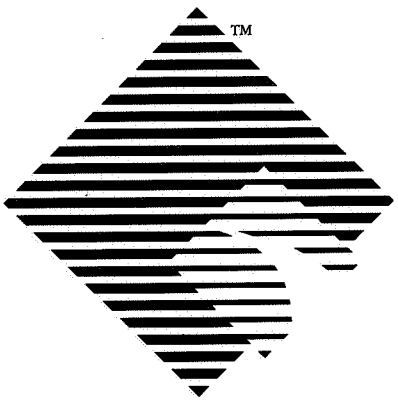
Key Specifications

	Workstations		Servers			
	DEC 3000 Model 400 AXP	DEC 3000 Model 500 AXP	DEC 3000 Model 400s AXP	DEC 3000 Model 500s AXP	DEC 4000 Model 610 AXP	DEC 7000 Model 610 AXP
Number of Processors	1	1	1	1	1-2	1-4
Cache Size	512KB	512KB	512KB	512KB	1MB	4MB
Performance (SPECmarks)	104	118	104	118	129	162
Storage Interface	SCSI	SCSI	SCSI	SCSI	SCSI/DSSI	SCSI/DSSI/CI

Alpha AXP specifications

Main technical characteristics for the first Alpha AXP implementation, the DEC 21064 chip:

- Multiple instruction issue, pipeline, 64-Bit, load/store, reduced instruction set architecture.
- The world's fastest IEEE compatible floating point chip.
- Dual instruction issue.
- CMOS (Complementary Metal Oxide Semiconductor) comprising 1.68 million transistors. Minimum feature size is .75 microns, transistor channel length is 0.5 microns and operating at 3.3 volts.
- Power dissipation 30 watts with 64-Bit virtual and physical addresses and 64-Bit integers and floating points, no operating system or language bias and four million times the addressable space of existing chips.
- Clock rates up to 200MHz are possible, and the chip is currently capable of delivering 400 peak MIPs and 200 MFLOPS.
- A complete CPU on a single chip with a transistor count of 1.68 million devices, including full integer and floating point execution units. These, together with related addressing and branching units, are fully pipelined and each is capable of launching a new instruction every cycle.
- Two high speed memory caches. An eight KByte instruction cache provides two 32-Bit instructions per clock cycle to the instruction despatch unit, and an eight KByte data cache can provide a 64-Bit data access during each cycle. This results in a cache bandwidth of 3.2 Gigabytes per second.



Alpha AXP

F A C T

Four million times the addressable space of 32-Bit architecture

Customer Services Supporting Alpha AXP

Digital is far more than a hardware and software supplier. It provides complete information systems - and has invested considerable time, effort and money to develop an organisation which can provide a comprehensive range of first class services.

In addition to having forty thousand people world-wide at four hundred and fifty locations in sixty-four countries, including more than three thousand committed to service in the UK, Digital provides an extensive range of management, business and support services.

AXP Services

- **Resource Centres Throughout Europe**

  To ensure that technical issues are dealt with quickly and effectively Digital has established a network of AXP Resource Centres in twenty different countries throughout Europe. These Resource Centres support application providers and end-users who require advice in moving their applications to the Alpha AXP architecture.

  These centres are staffed by technical consultants and are backed up by Central Resource Centres in Valbonne, France and Maynard, Massachusetts.

- **Porting Assessment Services**

  A consultancy to help customers to take advantage of the new AXP technology across their range of applications. It identifies technical dependencies (if any), sizes the effort required to move the applications and defines the sequence of the activities associated with the move.

- **Training**

  Designed to provide customers and partners with whatever knowledge and skills they need to take full advantage of Alpha AXP. Training covers a high-level overview of Alpha AXP architecture, microprocessor design and detailed device-driver and operating systems internals. There are separate or combined courses for OpenVMS, UNIX and Microsoft Windows NT users as well as for software and hardware developers.

- **Application Re-Engineering Services**

  These provide full support to customers who want to re-engineer applications currently running on non-Digital platforms, in order to take advantage of Alpha AXP openness and performance benefits. With Alpha AXP, Digital will be increasing its commitment to open systems technology consultancy, CASE consultancy and applications re-engineering.

Alpha AXP. The 64-Bit Future

With its primary advantages of speed and addressable space, Alpha AXP has massive potential for the future. New forms of user interface will become possible, changing the way in which we use computers. For instance voice, gesture and semantic recognition will become commonplace. Allowing even the non-computer literate business person to take advantage of new, super-powerful Alpha AXP based computers.

Systems based on Alpha AXP will also affect the performance of real-time applications such as flight and vehicle simulation, improving both the quality and depth of the simulation and the speed at which results can be obtained.

The Alpha AXP architecture's high speed mathematical computation ability will have a marked effect on everything from aeronautics to financial services to medicine.

Alpha AXP

F A C T

System design and architectural openness ensure a clear migration path and investment protection for the future