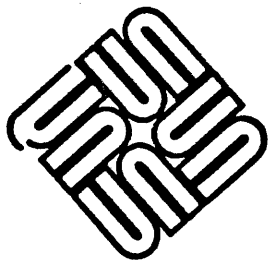


An 80 MHz Bipolar ECL implementation of SPARC

Anant Agrawal

Sun Microsystems, Inc.

2550 Garcia Avenue

Mountain View, CA 94043

ECL SPARC

- World's first single chip ECL 32 bit processor.
- World's fastest microprocessor. 80 MHz 12.5 ns cycle.
- 100% SPARC compatible.
- Substantially improved five staged pipeline.
- Demonstrates the scalability of SPARC.
- Full chipset available today.
- The ECL implementation consists of two chips.

  The integer unit (B5000)
  The floating point controller (B5100)

- These chips used in conjunction with the following BIT chips provide a complete SPARC integer and floating point solution:

  B5110, Floating Point Multiplier
  B5120, Floating Point ALU
  B5210, Register file.

- Both IU and FPC are full custom implementations in BIT P111 bipolar ECL process.

Integer Unit Features:

- Single chip implementation of SPARC Integer Unit architecture.
- Supports two co-processor interfaces.
  A dedicated Floating Point interface.
  A general purpose coprocessor interface.
- Has separate uni-directional data-in and data-out buses. Instructions and data are fetches over a single 64 bit bus. Data is sent out over a single word 32 bit bus.
- Supports prefetching and instruction buffering. This keeps pipeline full when executing memory reference instructions without requiring two caches.
- Pipelining supports single cycle loads, taken/untaken branches and FPOP dispatches. Stores take two cycles.
- Instruction/data buses and internal register file support parity.
- Synchronous traps are supported on parity error.
- Miss address is saved on the chip.
- Provides data and address passthrough mode to enable accessing the cache externally without putting additional loads on the cache buses.
- Supports seven register windows.
- Has matched clock and write enable delays for a more robust system design.
- Chip is partially scanable. Most of the data-path and all control latches can be scanned

Integer Unit

- The chip is divided into four major functional blocks.

  I/O interface unit.

  Register file: Consists of a 128*32b three ported RAM configured as 7 windows of 32 registers each. There is built-in redundancy in the RF to improve yields. On power-on a bad block of eight register can be configured out.

  Data path section: Consists of a 32bit Carry look ahead ALU, a 32 bit barrel shifter, address adder, pipeline registers and instruction buffers.

  Control section: Consists of controls for the processor.

- The IU is packaged in a 279 pin multilayer ceramic PGA.

Number of inputs	105
Number of outputs	109
Power and Ground	64

- 36 of the outputs are 30 ohm drivers.

IU Pipeline

- The IU uses a five stage pipeline:

  Fetch Stage:
  Instructions fetched from external cache into either the instruction register or instruction buffers.

  Decode Stage
  Instructions decoded
  Operands read from the register file
  Branch/Call target addresses computed

  Execution Stage
  Operands operated upon
  Data address generated
  Condition codes evaluated

  Memory Stage
  Data fetched from/dispatched to external cache.

  Write Stage
  Result written in the register file.

- This five stage pipeline is supported by three internal bypasses to reduce data dependent interlocks.

Major buses

Address Low (18:3)
Address Low~ (18:3)

Complimentary address buses used to address two banks of a caches. Can drive 30 ohm transmission lines. Registered outputs that are clocked off an early clock. providing more than one cycle for cache access.

Address high (31:15)

50 ohm bidirectional bus. Used to send the high part of the virtual address out. Can also be used as an input bus to address the cache externally. (Lets you limit the number of sources on the AL bus to one). The overlap in some of the AL/AH bits let you address 64KB- 512KB cache.

DIN (63:0)

Fetches two instructions or a double word of data into the IU. Instructions are either used right away or put in instruction buffers.

DINP(7:0)

Data parity associated with every byte of data in the DIN bus.

DOUT(31:0)

Used to transfer:

- Data out from IU
- Branch/Jump target addresses (to keep the PCs in the FP/CP units in sync with IU)
- FP/CP instructions.
- D Stage instruction address (Default)

ASI(7:0)

Address Space Identifiers. Used with the 32b virtual address to define 256 distinct address spaces.

Technology

1.2 Micron Bipolar ECL process.

3 layer metal interconnect. 4 Micron pitch for first and second layer metal. 8 Micron pitch for the third layer. All layer have 2 micron minimum metal line widths.

Chip Sizes:

IU: 9500 X 9800 microns
FPC: 9000 X 9000 microns

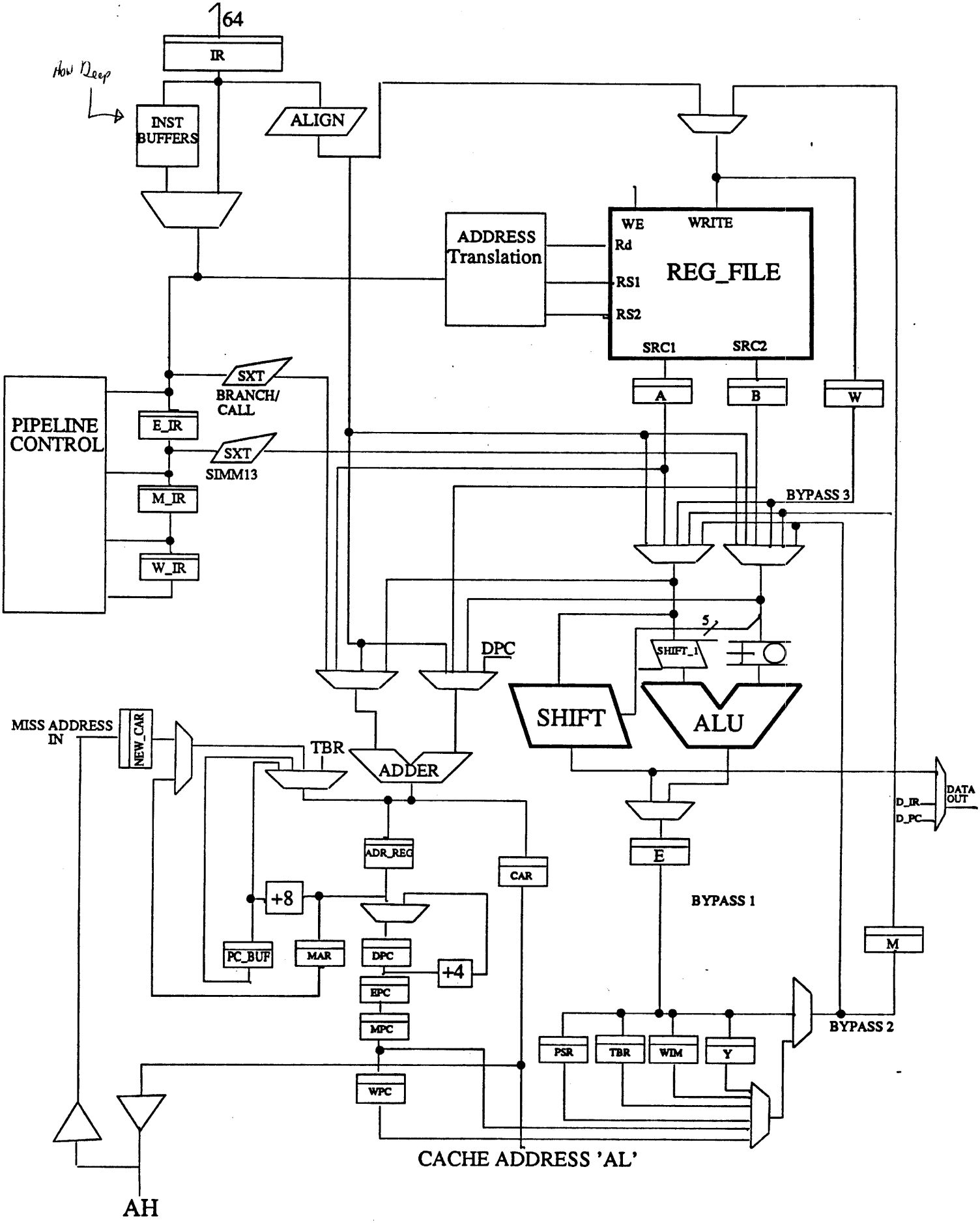
Number of drawn transistors:

IU: 125,000 transistors.
FPC: 36,000 transistors.

10 KH compatible I/O levels.

Extensive use of three level series gating in the internal logic.

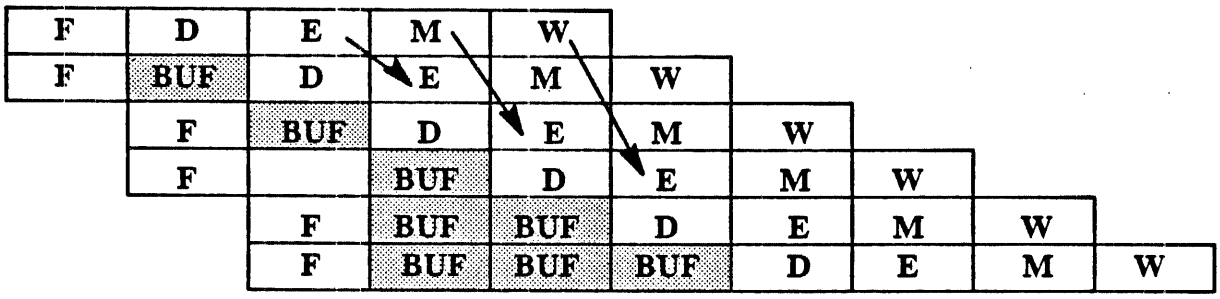
20 W



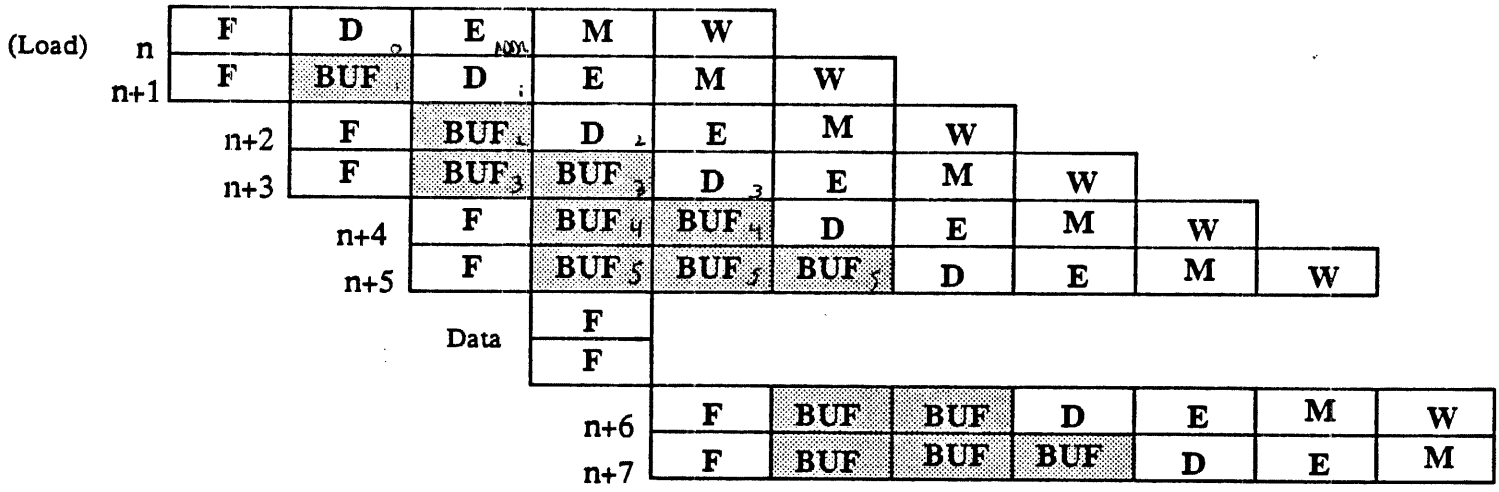
INTEGER UNIT

B5000 Pipeline Summary

- **Pre-Fetching, Queueing and Forwarding**

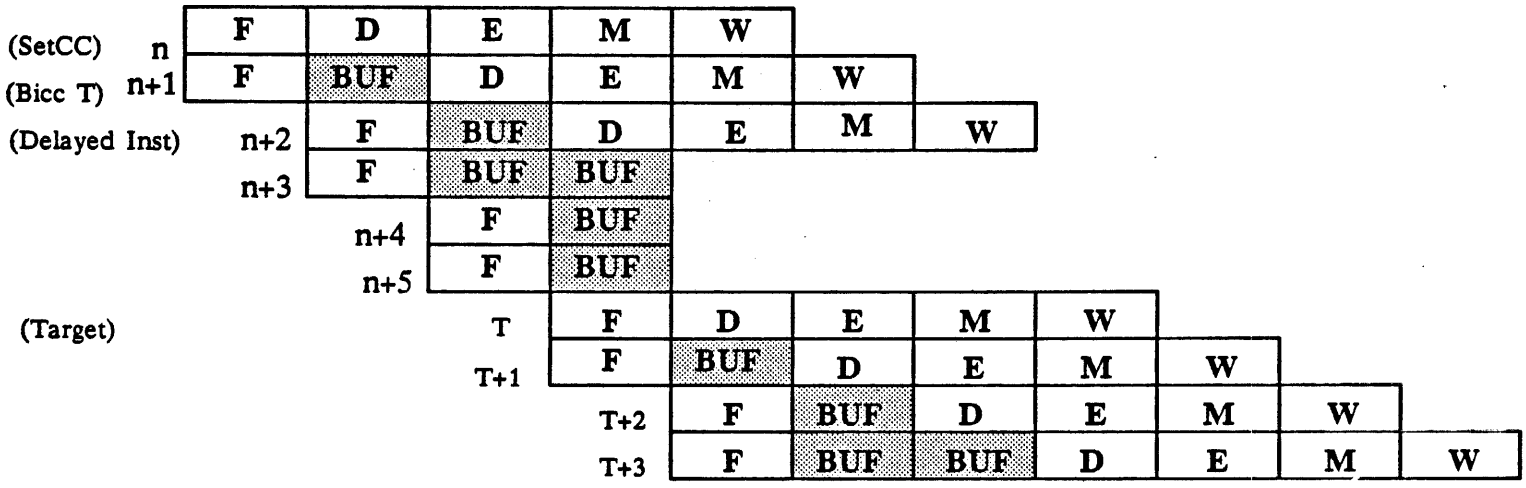


LOAD

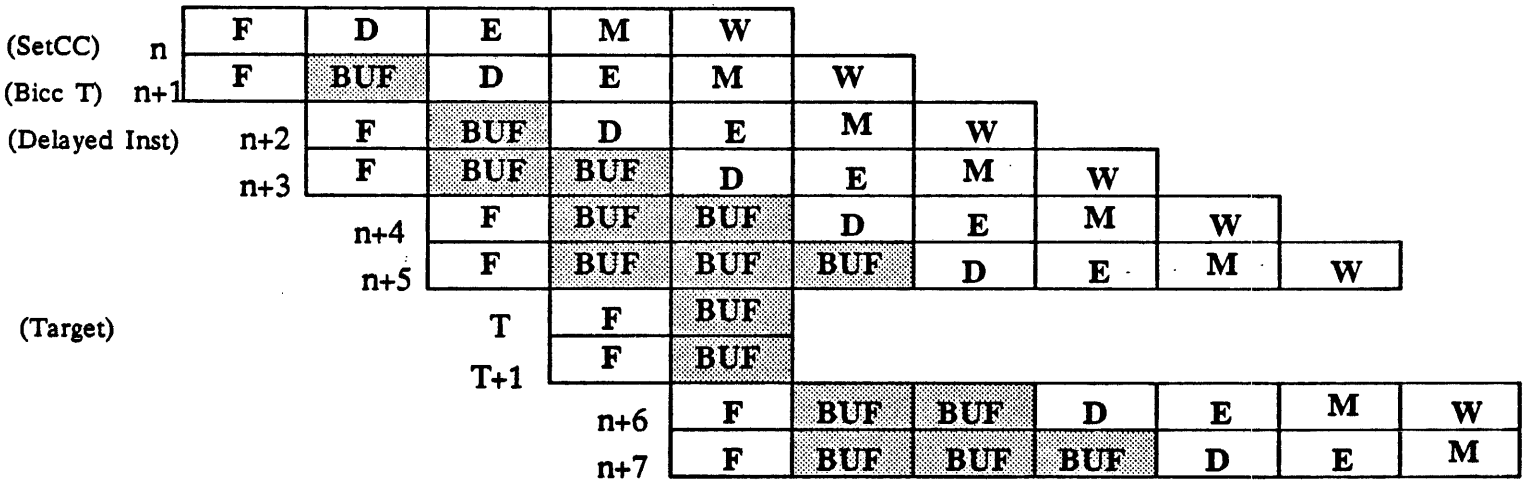


B5000 Pipeline Summary

TAKEN BRANCH



UNTAKEN BRANCH



Tradeoff: One cache vs Two

Issue: Which is better

Two separate caches each 32 bits wide.
One combined cache 64 bits wide.

Studies:

Single 64 bit cache is a better solution

Requires less number of pins on the processor (128 vs 160)

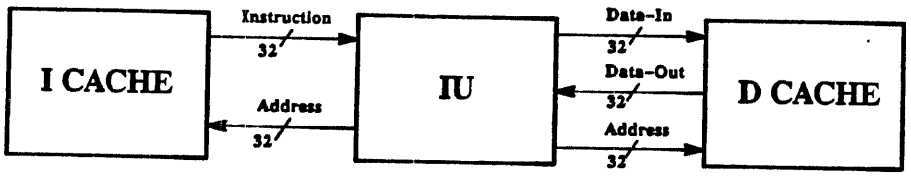
With an additional copy of the low part of the address, can access two banks of RAMs with shortened trace delays resulting in a faster cycle time.

Simpler system design (1 vs 2 caches and associated controls)

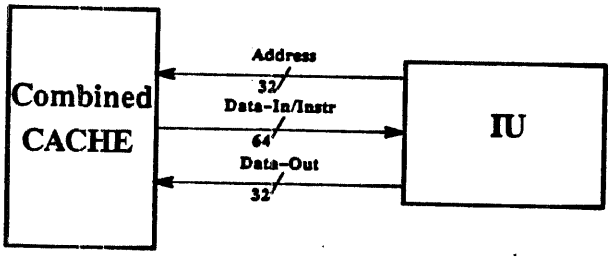
RISC statistics show small instruction buffers work well.

Minimal integer performance differences

Substantial better Floating Point performance (30% - 50%).



SEPARATE CACHES



Number of cycles per instruction

Instruction Type	# of cycles
ALU	1
Shift	1
Branch (taken)	1
Branch (untaken)	1
Integer Load Single	1
Integer Load Double	2
Integer Store Single	2
Integer Store Double	3
Floating Point Load Single	1
Floating Point Load Double	1
Floating Point Store Single	1
Floating Point Store Double	1
Floating point operations	1

Pipeline Interlocks:

- More than three load instructions in a row.
- Store instructions followed by a memory reference instruction.
- Branch followed by a store. Branch Add on Dont?
- Load data usage in the next instruction.
- FPOP at the target of a branch instruction.

System Design Considerations.

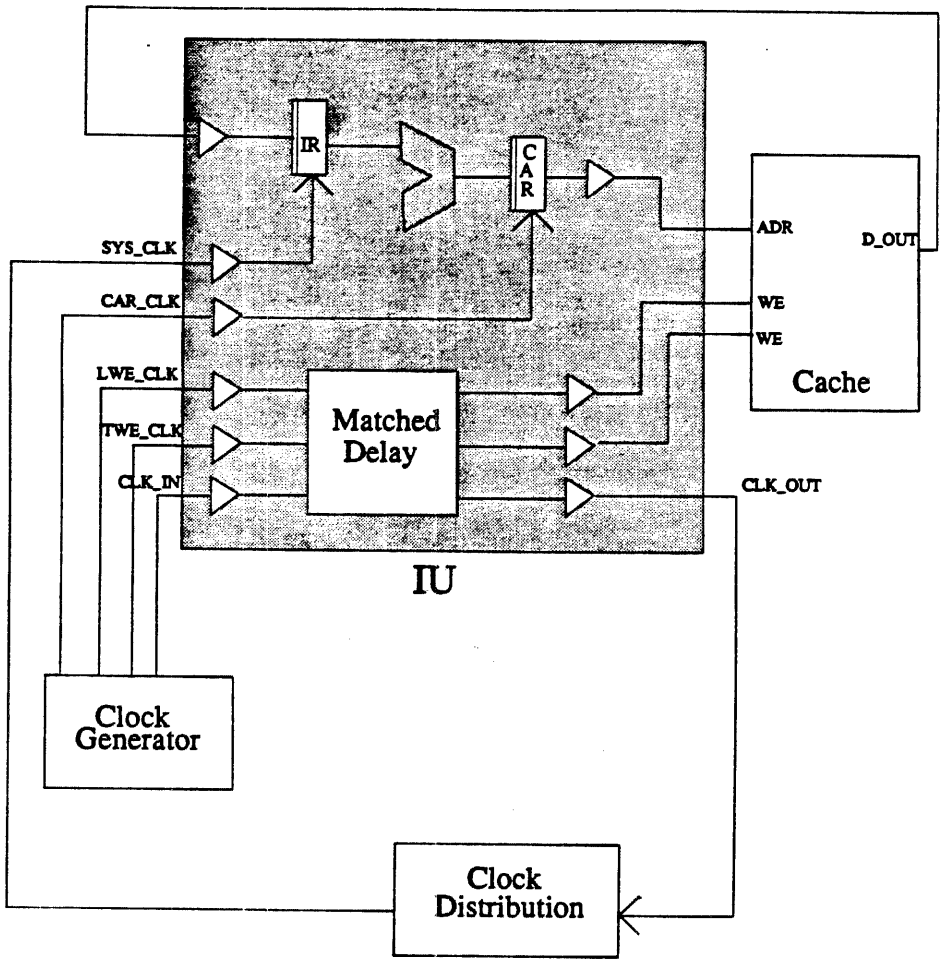
- Synchronous parity traps.
- support for large caches upto 512 K Byte.
- Differential AL bus:

  Two copies of the address bus enables splitting the cache in two banks. This reduces the trace delays resulting in a faster cycle time.

  30 ohm drivers enable placing memory chips closer together in a 50 ohm transmission line environment.

  Latched, deskewed address drivers eliminate the need for external drivers

  Early AL clock provides more time for cache access.
- Matched clock and write enable delays to ease system designs.
- All the buses (except the non critical AH bus) are unidirectional. This provides better transmission line environment.
- Bi-directional AH lines in passthrough mode let you address the cache from multiple sources with only one driver on the address traces.
- Double word cache data-path minimizes cache miss penalty.



Floating Point Controller

- Single chip implementation of SPARC Floating point controller.
- FPC controls separate MUL and ADD units and FP register file.
- Double word FP load and store buses.
- Entire floating point subsystem supports parity.
- Floating point status register on FPC.
- Four entry floating point queue; instruction address pairs of dispatched floating point operations.
- Both the FP execution units operate in parallel with the integer unit.
- All Sparc instructions supported except Extended precision operations.
  - fmul, fdiv, fsqrt
  - fadd, fsub, fconv and fmovs
  - floating-point load/store instructions
- IEEE-754 compatible
- FPC is completely scannable.

Floating Point Controller

- The FPC is packaged in a 259 pin multilayer ceramic PGA

Number of inputs	69
Number of outputs	100
Power and Ground	87

- 36 of the outputs are 25 ohm drivers.
- Dual ported floating point queue; allows independent and overlapped operand-fire and result-unload of functional units.
- FPC maintains own copy of the E-stage PC; allows single cycle dispatch of instructions.
- D-cycle opcode dispatch from IU. Operation started in function units before completion in IU pipe.
- Supports chaining where allowed.
- Single cycle FPreg loads and stores.

FP UNIT Major Buses

All output signals are driven from slave latches and all inputs are received in master latches.

ADROP(31:0)
Same as DOUT bus from IU, used to receive FP instructions/address and FSR load data from the IU to FPC.

LDATA(71:0)
Load data from cache to FP register file. Double word plus byte parity.

STD(71:0)
Store data bus from FP register file to cache or Source Operands bus from FP register file to FMUL and FALU. Double word plus byte parity.

RSLT(35:0)
Result data bus from FMUL and FALU to each other in the case of chaining or to FP register file. Single word plus byte parity.

MISC(35:0)
Bidirectional ECL bus. Used by FPC during store FSR or store FP queues.

Performance Example

cycles	instruction mix / basic cycle count	%
5,268,022	instructions	76.2
338,208	stores	4.9
143,007	JMPL	2.1
191,599	annulled instructions	2.8
373,085	load interlocks	5.4
159,747	Instr. Starvation	2.3
263,247	Cache conflicts	3.8
6,736,915	cycles, OR CPI = 1.28	

Bipolar Integrated Technology, Inc.

Advance Information

B5000
BIT SPARC Integer Unit

Features

- 32 bit high performance RISC processor
- Five stage pipelined instruction execution
- 80 MHz clock frequency
- 1.2 cycles per instruction average execution rate
- 50 CISC equivalent mips typical performance
- Separate input and output data buses
- Double word instruction/data input bus
- 32 bit addresses provide access to 4 Gbytes
- 8 bit Address Space Identifier (ASI) defines 256 alternate address spaces
- Optional byte parity checking on input bus and internal register file
- 120 internal registers configured as 7 windows
- All instructions are 32 bits wide
- Floating point coprocessor support included
- Additional user-defined coprocessor interface
- ECL 10KH compatible I/O
- 279 pin PGA package

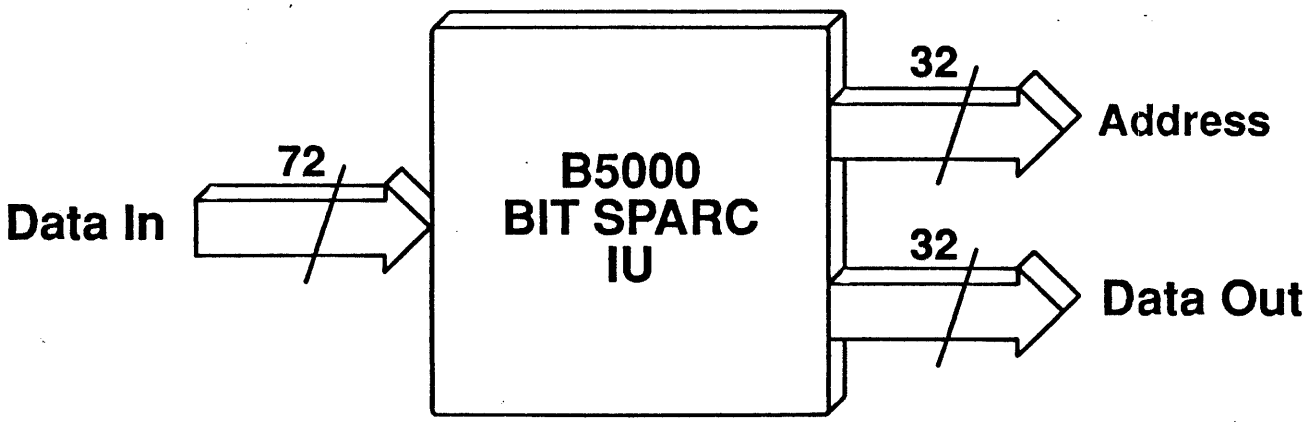
Description

The BIT SPARC™ Integer Unit (IU) is an ECL VLSI implementation of the SPARC microprocessor architecture. Implemented in a 2 micron ECL technology, BIT SPARC provides users with a high end upgrade path for SPARC implementations. Fully compatible with the SPARC architecture, BIT SPARC achieves nearly one cycle per instruction.

Maximum data throughput is achieved using a five stage pipeline and separate load and store buses. Additionally, the input bus used for fetching instructions and loading data operands is double wide, allowing two instructions to be fetched in a single cycle. Most buses are uni-directional, easing the system designer's task of creating controlled impedance ECL environments. Further, the output drivers for the low order addresses are designed for 30 ohm impedances, matching the requirements of printed circuit traces with heavy capacitive loads.

The BIT SPARC IU features a large register file configured in a circular-set-of-windows structure. Procedure calls can be optimized by switching windows, thus minimizing the need for external memory accesses. ALU operations are executed in a register to register fashion, requiring just one execution cycle of a pipelined operation. Bypass paths are utilized to fully interlock the pipeline. An 8-bit Address Space Identifier (ASI) allows supervisor mode distinction of 256 separate address spaces.

Two independent coprocessor ports are included, one for the floating point subsystem and the other to be defined by the user. Parity is supported on the input data path, as well as the internal register file to increase system reliability. Aimed at air cooled applications, BIT SPARC is a monolithic solution to high performance needs.



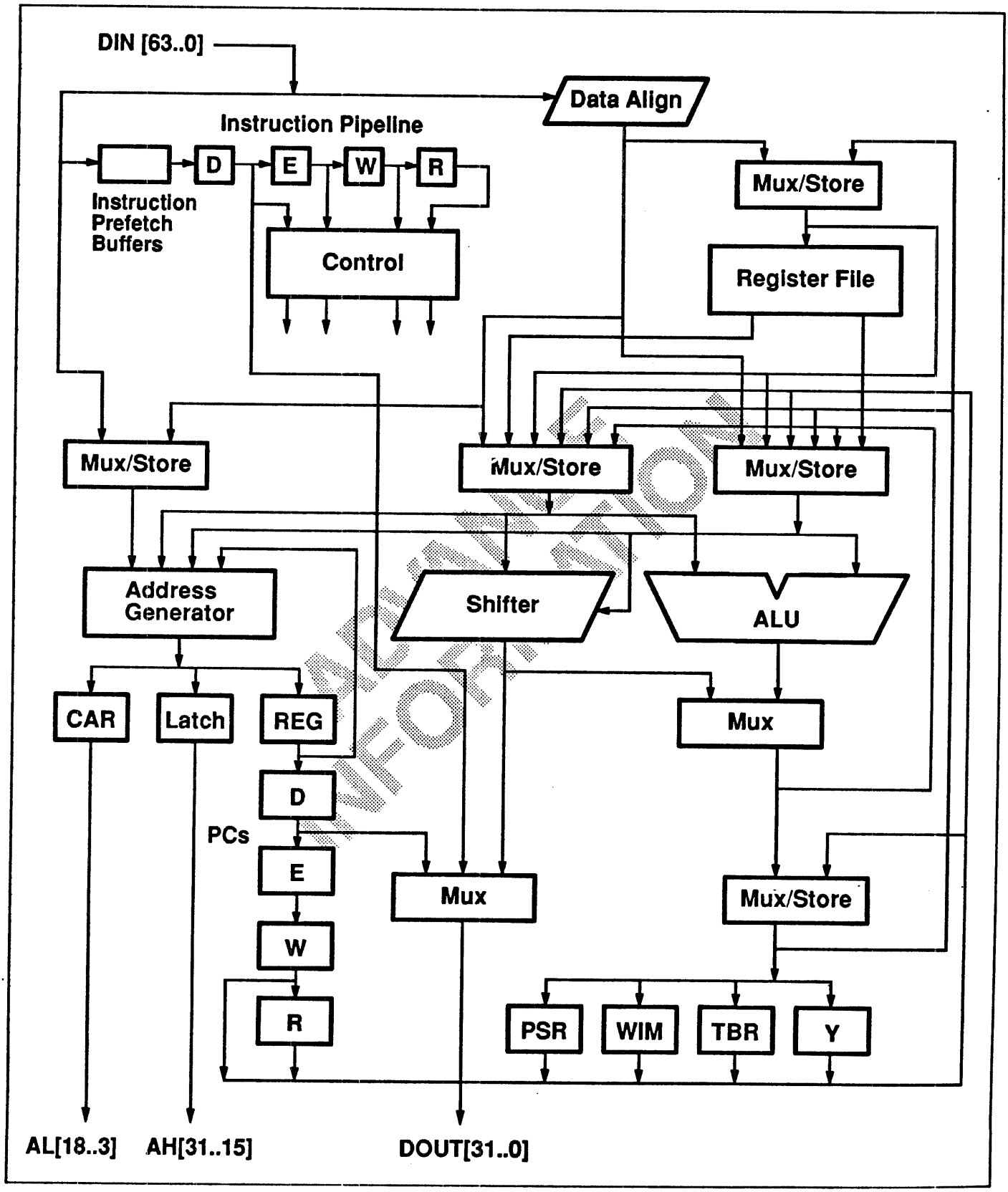


Figure 2 — BIT SPARC Integer Unit Block Diagram

Functional Description

Processor Overview

BIT SPARC is a VLSI processor based on Reduced Instruction Set Computer (RISC) architecture. All instructions are 32 bits wide. Designed to provide a high end upgrade path to users of the SPARC architecture, it offers extremely high performance at a low cost per mips (million instructions per second). BIT SPARC 32 bit addresses and 8 bit Address Space Identifier provide addressing of up to 256 pages of 4 Gbyte of virtual memory.

Functional blocks of the IU consist of: a very high bandwidth bus interface (640 Mbytes/second peak), an integer execution unit which includes an ALU and a barrel shifter, an address generator operating concurrently with the execution unit, a three-port register file of 120 registers, a four place internal instruction buffer, a five stage instruction pipeline, and two coprocessor interfaces for floating point and a user definable function.

Bypass paths and complex control logic fully interlock the internal pipeline and achieve nearly single cycle average execution for a typical application.

Important Features

Single cycle execution — most instructions execute in a single cycle when operating out of cache.

Pipelined execution — due to the pipelined architecture of the IU, execution of instructions is completed in stages. Five stages of the pipeline typically contain five separate instructions in various stages of completion. The result is a peak rate of one operation per cycle.

Simple instruction format — All instructions are 32 bits wide and perform simple operations.

Register to register architecture — all ALU operations are performed on register (or immediate) operands and placed back into the register file. Only load and store instructions access memory.

Large register file — 120 registers are included on chip; subroutine calls and parameter passing are performed using registers. This reduces the need to access memory, which is the typical method in a stack based machine.

Delayed control transfer — except in special cases such as traps, the IU executes the instruction immediately following a control transfer instruction. An annul bit may be set in branch instructions which cancels the following instruction if the branch is not taken, increasing the probability that an optimizing compiler can place a useful instruction in the delay slot.

Hardwired control — by keeping the instruction operation simple, the need for an internal micro-coded engine such as those found in CISC architectures is eliminated.

Two coprocessor interfaces — floating point coprocessor operations are supported in the standard instruction set. A separate set of coprocessor operations can be defined by the user, provided a second coprocessor has been defined.

Double word instruction bus — by using a double word instruction bus the BIT SPARC IU is capable of fetching two instructions simultaneously. One or the other of these instructions can be used immediately.

The additional instruction(s) can be placed in an internal buffer for temporary storage until needed or discarded. Four places deep, the instruction buffer reduces the likelyhood of pipeline stalls waiting for instruction fetch.

Internal Pipeline

Fast cycle times, resulting in very high performance, are possible due to the architecture of the instruction execution unit. Five stages of pipelining are used to sequence through operations, Fetch, Decode, Execute, Memory Access, and Result as follows:

Fetch (F) — the fetch stage is the cycle required to fetch instructions from the bus and clock them into the internal instruction buffer.

Decode (D) — instructions are clocked into the decode unit from the instruction buffer for translation and generation of internal control signals and sequencing; alternatively, they may be taken directly from the bus.

Execute (E) — after the instruction is decoded, the register (or immediate) operands are clocked into the execution unit (ALU, barrel shifter, address generator) and the operation is performed.

Memory Access (W) — loads use the W cycle to perform the cache access in a cache based system. This allows the data to be available for latching into the input port at the end of the W cycle. For ALU operations, the W stage is a one cycle delay.

Result (R) — loaded data or computed results are written to the register file during the R stage.

The decode unit is aware of data dependencies and other situations which cause pipeline stalls (one stage of the pipe awaiting the completion of another stage). Due to the fully interlocked pipeline, stalls only occur in the BIT SPARC IU when there is external bus contention or when an external access requires more than a single cycle to fully complete. Bypass paths and control logic are in place to account for and remove internally generated pipeline stalls.

Delayed Control Transfers

Instruction sequencing is performed by fetching instructions from a series of sequential addresses and executing the instructions until a condition occurs which forces execution to resume at a non-contiguous location. Control transfer instructions are operations which, when executed, present such a condition. These instructions include calls, branches and jumps.

Pipelined systems typically prefetch instructions so they will be available after the current instruction executes. Pipelining hence presents a problem when the current instruction is a control transfer, since the next sequential instruction in memory will also be next in the pipeline to be executed at the time the target instruction is being fetched. Two things can happen; the pipeline can stall waiting for the target instruction to be fetched,or the sequential instruction can always be executed before transferring control. Stalling the pipeline is inefficient and forces all control transfer instructions to require two cycles minimum. A typical application will contain about 20% control transfer instructions, resulting in noticeable performance degradation. Since a useful instruction can often be placed in the delay slot (ie — the next sequential place in memory) by an optimizing compiler, some of the control transfer performance can be bought back if this delay slot instruction is always executed. This method of always executing the next sequential instruction is called delayed control transfer architecture.

BIT SPARC uses a delayed control transfer architecture. By assuming control will be transferred, the IU will start fetching the target instructions. If the optimizing compiler has been successful in placing a useful instruction in the delay slot, no cycles will be lost. In addition, many instructions for control transfer include an annul bit, which can be set to force the IU to cancel the delay instruction when a conditional branch fails. This provides more freedom for the compiler to place a useful instruction into the delay slot, since an instruction with the annul bit set need only be useful in the case of a taken branch.

Traps

Two types of traps are recognized by the BIT SPARC processor; synchronous, software controlled traps and asynchronous, interrupt controlled traps. Coprocessor traps are synchronous since they only occur on coprocessor instructions.

A table in memory will contain instructions which "vector" to trap handlers. Traps transfer control to these instructions using a base address and offset given in the Trap Base Register (TBR). The trap base address (the address of the trap instruction table) is controlled in the TBR by software, and the offset is controlled by the IU depending on the trap type. Only the reset trap doesn't use the trap table, but instead always starts executing instructions at absolute address 0.

Traps save the program counter and next program counter, either of the instruction which caused a synchronous trap or the instruction that was in the E stage of the pipeline when an interrupt or coprocessor trap occurred. In the case of floating point or coprocessor traps, the address of the instruction which caused the trap will be in the floating point queue.

Floating Point and Coprocessor Interfaces

The BIT SPARC architecture provides two separate coprocessor ports, a floating point coprocessor which has been defined and included in the language support, and a separate, user defined coprocessor. The IU fetches and dispatches coprocessor instructions from the same stream as the integer instructions.

Two classes of coprocessor instructions are included, coprocessor loads/stores and coprocessor operate. In the case of the floating point unit, the floating point operate instructions have been fully defined. It is up to the user to define the separate coprocessor operate instructions. The IU executes coprocessor/floating point loads and stores by generating the address and control signals required for reading/writing memory. A path is assumed from cache or main memory directly to the coprocessor registers, thus bypassing the IU. The execution of the operate instructions involves dispatching both the instruction as well as the instruction fetch address to the coprocessor. The coprocessor then completes the instruction concurrently with the IU operation.

The floating point or coprocessor unit consists of four functional blocks; a register file, an instruction queue which holds instruction/address pairs, a state register, and an execution unit.

BIT SPARC supports a floating point sub-system through the use of the BIT B3110A/B3120A floating point chip set and B3210A register file. A separate floating point controller interfaces the chip set to the IU. See related documents for information pertaining to these components.

Internally Generated Errors

Parity is checked by the IU on instructions and data loaded into the data input port, and on operands read from the internal register file. Whenever a parity error occurs, the IU generates a synchronous trap. These traps can be enabled using a bit in the Bootmode Configuration Register. Odd parity is used.

Additionally, the IU may encounter a situation which causes an irrecoverable error (for example, a trap occurs with traps disabled). In this case, the processor enters a halt state and must be reset to resume. This condition is indicated with an external signal.

Double Word Instruction Fetch

A typical BIT SPARC system will utilize a double word wide instruction/data cache connected to the input bus. Two advantages are gained by this architecture: 1) branches can be performed in one cycle since the IU will have both the buffered sequential instructions as well as a cycle to fetch the target instruction; 2) floating point performance is increased because double precision operand load and store operations can occur in one cycle. All pre-fetching and control on the double word instruction bus is handled by the IU. Internal state machines are responsible for keeping the instruction buffer in a non-empty state. During a branch the IU will fetch the target instruction and, when the branch condition is decided, will have both the target instruction and the next sequential instruction available. Consequently, most branches will operate in one cycle.

General Purpose Registers

The register file in the IU includes 120 general purpose registers. These are configured as 7 windows of 24 registers each and 8 *global* registers. Windows overlap with the previous and the next window by 8 registers, leaving 8 registers which are *local*. Those registers shared with the previous window are the *ins*, where parameters are received from the calling subroutine. Registers shared with the next window are the *outs*, where parameters are passed to the called subroutine. Thus the current window's *ins* are the previous window's *outs*, and the current window's *outs* are the next window's *ins*. *Local* registers are private, and *global* registers are equally available to all.

Window management is provided using the Current Window Pointer (CWP) in the Processor State Register, save and restore instructions which decrement and increment the CWP, and the Window Invalid Mask register which places bounds upon the windows. Addressing of the window registers is performed using 5-bit fields in the instructions in conjunction with the 5-bit Current Window Pointer. Register numbers take values from 8 to 31, global registers are absolute numbers 0 to 7, and the Current Window pointer ranges from 0 to 6. The following table shows the register addressing.

Number	Name
r[24] to r[31]	ins
r[16] to r[23]	locals
r[8] to r[15]	outs
r[0] to r[7]	globals

Figure 3 describes register sharing across three windows.

Window numbers can be changed by altering the CWP field of the Processor State Register (PSR). The active window is defined as the window currently pointed to by the CWP. The CWP can be written by a WRPSR instruction, incremented by a RESTORE or RETT instruction, and decremented by a SAVE instruction. When CWP is decremented or incremented, modulo seven arithmetic is used since only seven register windows are available. Thus windows are joined together in a circular set. This makes window 6, the highest numbered window, adjacent to window 0, the lowest numbered window, with window 0 *outs* forming window 6 *ins*.

Windowed register addressing also uses the Window Invalid Mask (WIM) register for window maintenance. This register contains a control bit for each window. If a SAVE instruction encounters a "1" in the WIM bit corresponding to the new window, a register file overflow trap will occur. In the same way, a RESTORE instruction will cause a register file underflow trap if it encounters a 1 in the WIM bit corresponding to the new window.

Most of these registers are for general purpose use, but there are a few special usage cases. If global register r [0] is addressed as a source operand, the operand value 0 is returned. If r [0] is addressed as the destination operand, no register is modified. The CALL instruction writes its own fetch address into register r [15] of the current window, which is an *out* register and will be an *in* register of the called routine. Traps save the program counter [PC] and next program counter [nPC] into r [17] and r [18] of the next window, which will be local registers in the trap routine.

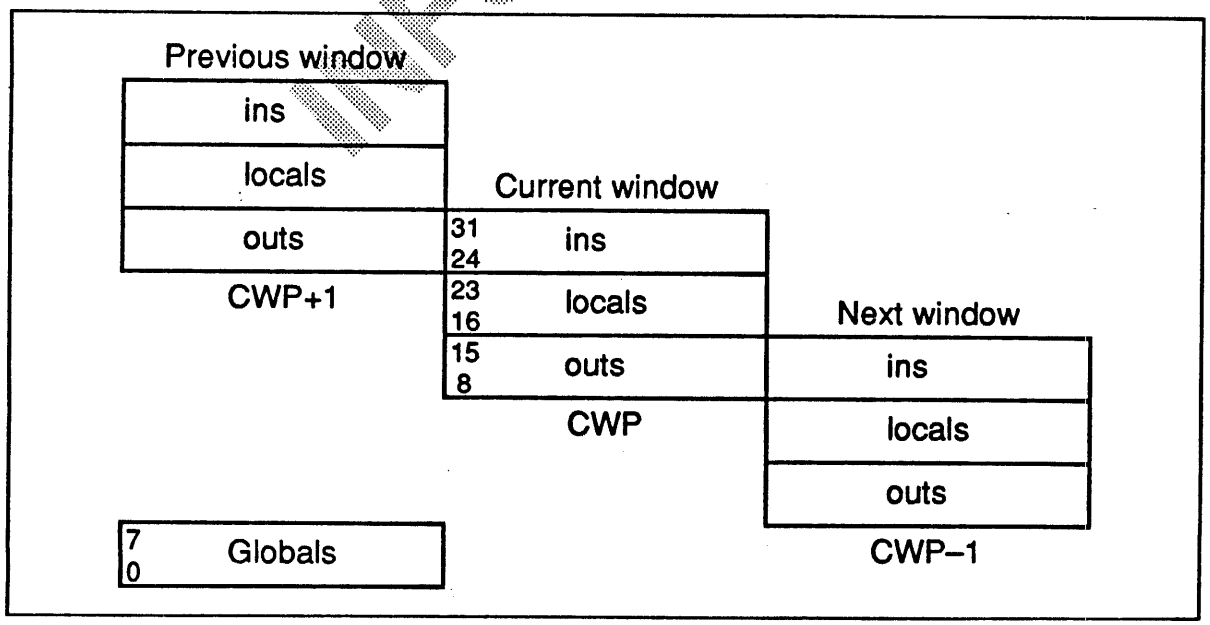


***Figure 3** — Register Windowing in the BIT SPARC*

Special Purpose Registers

Four special purpose registers are available to the programmer as 32 bit read/write registers. These registers are used for control and status information in most cases. They include the Processor State Register (PSR), Window Invalid Mask register (WIM), Trap Base Register (TBR), and the multiply step register (Y).

Processor State Register

This 32 bit register contains various fields describing mode and state of the processor. Two instructions, RDPSR and WRPSR, access the register directly, whereas some fields within the register may be modified by various instructions. In particular, the CWP field is modified by SAVE, RESTORE, Ticc, and RETT, and the integer condition codes are modified by several ALU instructions. Figure 4 shows the different fields in the PSR.

Impl — Bits 31..28 identify the implementation number of the processor. WRPSR will not change this field; for BIT SPARC, RDPSR returns the number 2.

ver — Bits 27..24 of the PSR identify the version number of this particular implementation. WRPSR will not change this field, and when read by RDPSR BIT SPARC returns it's version number.

Icc — Bits 23..20 of the PSR contain the integer unit's condition codes. These bits will be modified by WRPSR as well as several ALU instructions. Bicc and Ticc instructions base their control transfer on these bits. Their meanings are as follows:

Bit 23 — Negative; a "1" in this bit indicates that the result of the last ALU operation which modified the icc field was negative.

Bit 22 — Zero; when high, this bit indicates that the result of last ALU instruction which modified the icc field was zero.

Bit 21 — Overflow; when high, indicates that the last instruction to modify icc resulted in an arithmetic overflow. Logical instructions which modify icc always reset this bit.

Bit 20 — Carry; when bit 20 is high, the last ALU instruction to modify icc caused a carry out of bit 31 of the result if the operation was addition, or caused a borrow into bit 31 if the operation was subtraction. Logical instructions which modify icc always reset this bit.

reserved — Bits 19..14 are reserved for future use. WRPSR should only write zeros to this field; RDPSR will read zeroes from this field.

EC — Bit 13 enables the user defined coprocessor interface when high. See the SPARC architecture manual for further information.

EF — Bit 12 enables the floating point coprocessor interface when high.

PIL — Bits 11..8 determine the interrupt mask level. Interrupt priority is highest to lowest from 15 to 1. The processor will only accept interrupts whose level is higher than the value in the PIL field of the PSR. Interrupt level 15 cannot be masked.

S — Bit 7 determines the execution level of the processor. If S=1 the processor operates in supervisor mode, and when S=0, the user mode. Note that since WRPSR is a supervisor level instruction, supervisor mode can only be entered from user mode through the use of a trap.

PS — Bit 6 contains the value of S at the time of the most recent trap.

ET — Bit 5 is the trap enable bit, enabling traps when ET = 1.

CWP — Bits 4..0 are the current window pointer; used in conjunction with the relative register address in the instruction, CWP helps determine which absolute register is being addressed. CWP is decremented by traps and SAVE (ie. CWP "grows" down) and incremented by RESTORE and RETT instructions. Since the CWP cannot point to an unimplemented window, arithmetic is performed modulo 7.

Window Invalid Mask Register

The WIM register is used to determine if a window overflow trap or a window underflow trap should be generated by a SAVE, RESTORE, or RETT instruction. Each of the seven least significant bits in this 32-bit register corresponds to a general purpose register window. For example, bit 0 corresponds to window 0, bit 1 to window 1, and so on. Bits 7 through 31 correspond to unimplemented windows and read as zeroes. They are ignored when written.

When a SAVE, RESTORE, or RETT would cause the CWP to point to a window whose corresponding WIM bit is one, it causes a window overflow trap (SAVE) or window underflow trap (RESTORE, RETT) to occur.

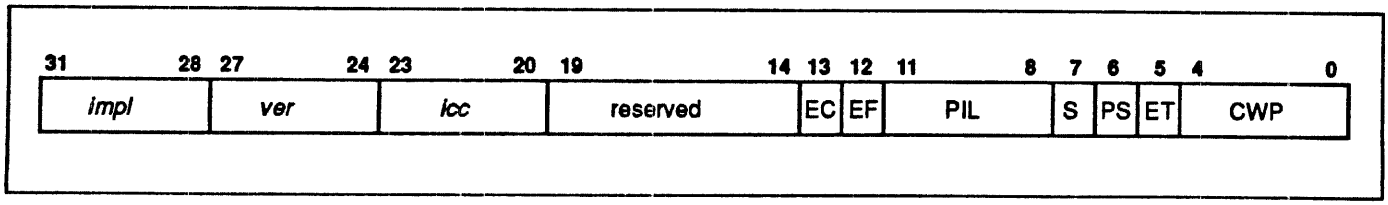


Figure 4 — BIT SPARC Processor State Register (PSR)

Trap Base Register

The Trap Base Register contains three fields which determine the address of a trap handler when a trap occurs. The fields are as follows. See Figure 5.

TBA — Bits 31..12 comprise the Trap Base Address, controlled by software. It contains the most significant 20 bits of the trap table address (except for the reset trap, which always vectors to zero). TBA can be written by using the WRTBR instruction.

tt — Bits 11..4 comprise the Trap Type field. This is an eight bit field written by the processor at the time a trap is taken, and retains its value until the next trap. Trap type is concatenated with the TBA to form the address of the first trap instruction; TBA is the base address and tt is the offset into the trap table. WRTBR does not affect the tt field.

zero — Bits 3..0 are always zero. WRTBR does not affect this field. Every trap address will have the least significant four bits zero, thus trap vectors are aligned on 16 byte boundaries.

Y Register

The multiply step instruction uses this 32 bit register to create 64 bit products. This register can be read and written using the RDY and WRY instructions. Consult the SPARC architecture manual for more information concerning the use of this register.

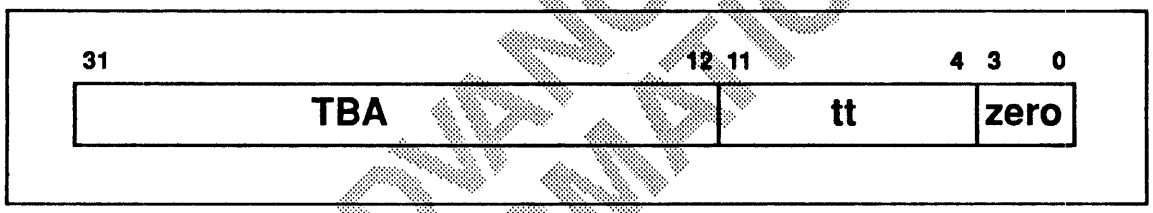


Figure 5 — BIT SPARC Trap Base Register

Processor Data Types

The BIT SPARC Integer Unit recognizes nine different data types, six integer and three floating point. The six integer data types include byte, unsigned byte, halfword, unsigned halfword, word, and unsigned word. Although the instructions include load (and store) double word, this is not considered a processor data type since the IU can only operate on up to 32-bit word operands; consequently, a double word would be operated on one 32-bit word at a time. A byte is 8 bits wide, a halfword 16 bits, a word 32 bits, and a double word is 64 bits. The ANSI/IEEE 754-1985 floating point data types include single, double, and extended. See the BIT SPARC Floating Point Controller documentation for the for information on the floating point data types.

The SPARC architecture defines byte order in memory to be most significant to least significant. For example, a double word in memory will have bits 32 to 63 placed in location n and bits 0 to 31 placed in n+4. Load double instructions will thus load the most significant 32 bits of a 64 bit operand from location n into register r specified in the rd field of the instruction, and the least significant 32 bits will be loaded from n+4 into register r+1.

This rule also applies to bytes or halfwords within a word. A word made up of four bytes n to n+3 will contain the most significant byte at byte address 0 and the least significant byte at byte address 3. The address of a double word, word, or halfword is the address of its most significant byte.

In addition, halfwords, words, and double words must all be evenly aligned. The address of a halfword must be evenly divisible by 2, that of a word by 4, and the address of a double word must be divisible by 8. If a load or store instruction generates an improperly aligned address, a *memory address not aligned* trap occurs.

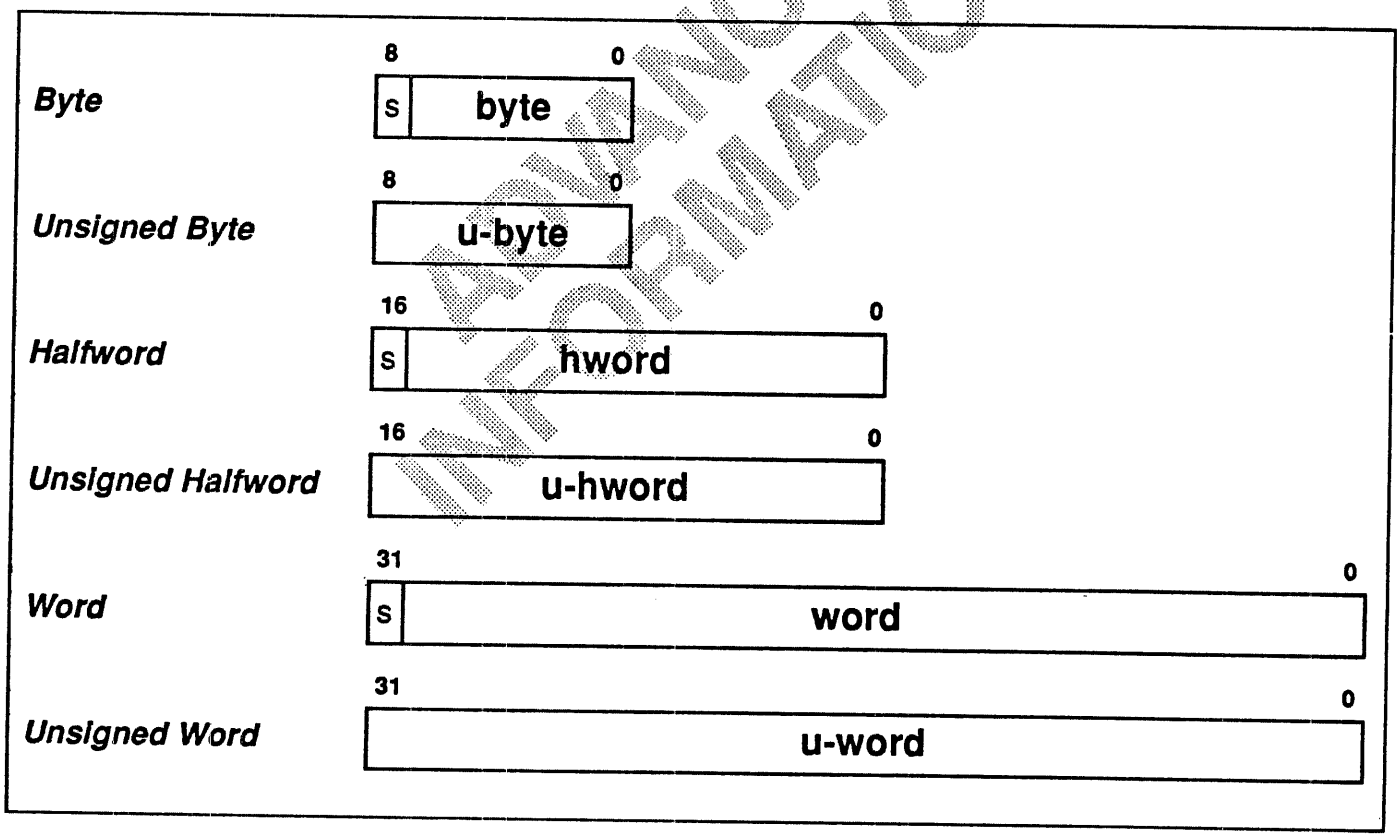


Figure 6 — BIT SPARC Integer Data Types

Instruction Set Overview

BIT SPARC instructions fall into six different categories: *loads and stores*; *arithmetic/logic/shift*; *control transfer*; *read/write special registers*; *floating point operate (FPop)*; and *coprocessor operate (CPop)*. These instructions are coded based upon three different formats. Format 1 has a 2-bit op field and a 30-bit displacement value. Only the *CALL* instruction uses format 1. Format 2 includes the same 2-bit op field, a 3-bit op2 field, a 22-bit displacement or immediate value, and either a 5-bit destination register field or a condition/annul field for branching. *Branch* instructions and *SETHI* use format 2. Format 3 is used by all other instructions. It has the same 2-bit op field, a destination register field rd, a 6-bit op3 field, a source register field, and 1 of 3 sub-formats: 1) *floating point* and *coprocessor operate* instructions contain a 9-bit *FPop (CPop)* field and a second 5-bit source field; 2) a 13-bit immediate data field; or 3) a second source field and an ASI field. More information concerning these formats may be found in the SPARC architecture manual. Figure 7 illustrates these formats.

Loads/Stores — These are the only instructions which access memory. They generate a 32-bit logical address and an 8 bit Address Space Identifier. Two methods of calculating an address are possible; an address can be obtained by adding two registers, or by adding one register and a 13-bit sign extended immediate value.

The SPARC architecture defines four of the 256 available address spaces to be *user instruction* (08H), *user data* (0AH), *supervisor instruction* (09H), and *supervisor data* (0BH). All other spaces are implementation/user definable. However, only supervisor level instructions may selectively place a value into the ASI field. All user instructions will automatically generate either 08H or 0AH.

Arithmetic/Logic/Shift — These instructions compute a result that is a function of two source operands, and either write the result into a register or discard it. The two operands are either registers (i=0) or one register and a 13-bit sign extended immediate value (i=1). Most of these instructions have a dual version which modifies the integer condition code field in addition to performing the operation. A result is discarded by writing it to global register 0.

These instructions perform *integer arithmetic*, *logical*, or *shift* operations. None of the *shift* instructions change the condition codes. In addition, there are tagged arithmetic instructions which set a condition code if either of the two least significant bits of either operand is 1.

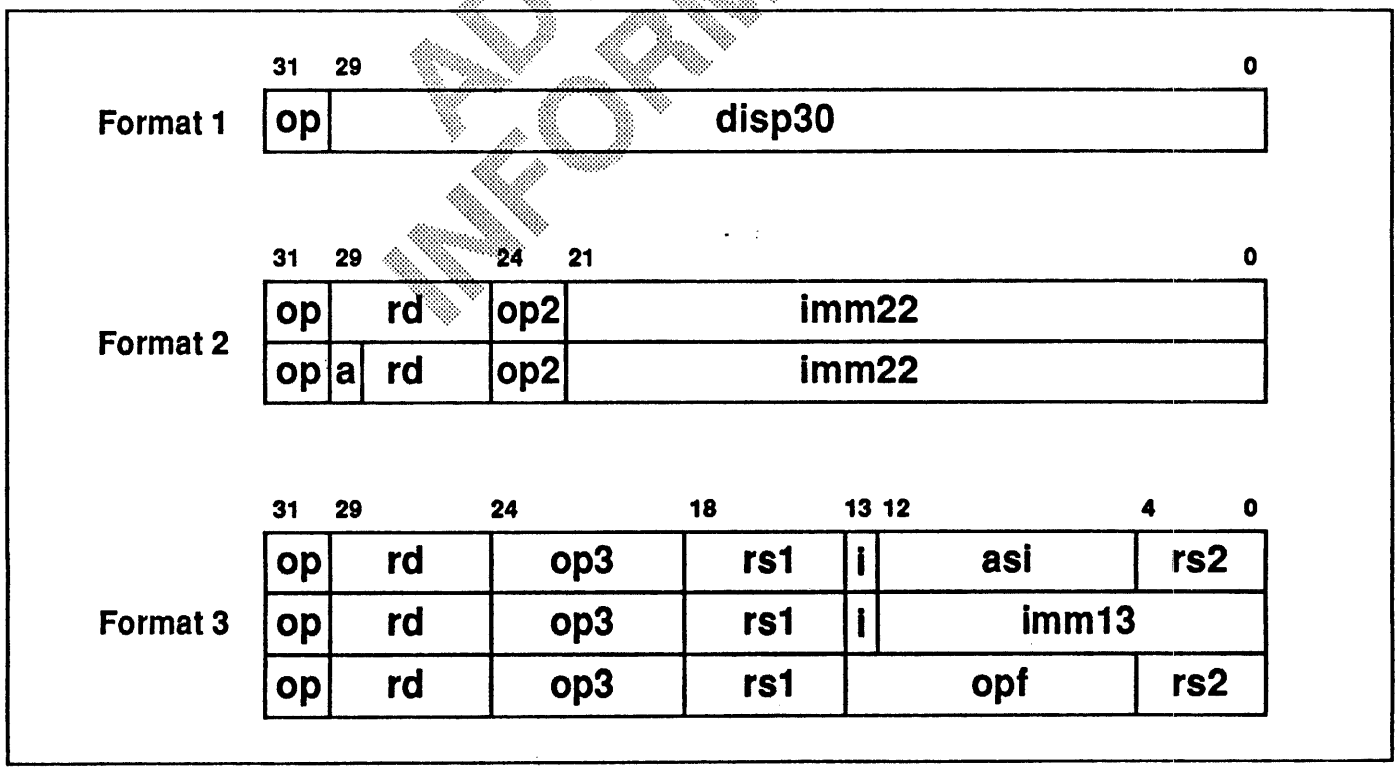


Figure 7 — BIT SPARC Instruction Formats

Control Transfer Instructions — These instructions include *conditional branches, jump and link, call, synchronous trap,* and *return from trap.* They can be characterized as instructions which change the value of the program counter (PC) and next program counter (nPC). All of these instructions transfer control to a non-contiguous location in memory, and all but the *synchronous trap* execute the instruction immediately following sequentially (ie — delayed control transfer).

Branch and *call* instructions use program counter relative displacement, that is they add an immediate value to the current value of the program counter. Branches use a 22-bit displacement, allowing transfer of control within 8 Mbytes; *call* instructions use a 30-bit displacement which allows transfer to an arbitrary location.

The *jump and link* and *return from trap* instructions use register indirect displacement. They compute their target addresses as either the sum of two registers (i=0) or the sum of a register and a sign extended immediate value. Trap instructions always use the Trap Base Register to calculate their trap address.

Read and Write Special Registers — The SPARC architecture provides instructions to read and write the contents of the various programmer visible special registers. The source or destination is implied by the instruction itself. These include instructions to *read and write the PSR, the WIM, the TBR, and the Y register.* Only the Y register may be read and written in user mode. All other read/write special register instructions are privileged. When the signal BOOTMODE is high, the WIM register instructions can be used to gain access to the Bootmode Configuration Register.

Floating Point Operate — *Floating point operate* instructions execute concurrently with the IU instructions. They are generally three register instructions which place their results in a destination floating point register, the result being a function of one or two source operands. The exception is *floating point compare*, which modifies the condition code field of the floating point state register. *Floating point operate* instructions do not include *floating point load and store* instructions.

Because the IU and FPU operate concurrently, at the time of a floating point exception the IU program counter might not contain the address of the floating point instruction that caused the exception. However, the first element of the floating point queue contains this instruction and it's address, and the rest of the queue contains instructions and addresses which have not yet completed. These may be re-executed or emulated in software.

Coprocessor Operate — *Coprocessor operate* instructions, along with *coprocessor load and store* instructions, utilize an interface separate from, but identical to the floating point interface. Like the floating point operate instructions, these specify two source operand registers and a destination operand register. A user defined coprocessor using this interface may also operate concurrently with the IU.

Table 1 — Instruction Set

Mnemonic	Function	Cycles	Format
LDSB	Load byte	1	F3
LDSH	Load halfword	1	F3
LDUB	Load unsigned byte	1	F3
LDUH	Load unsigned halfword	1	F3
LD	Load word	1	F3
LDD	Load doubleword	2	F3
LDSBA	Load byte from Alternate space	1	F3
LDSHA	Load halfword from Alternate space	1	F3
LDUBA	Load unsigned byte from Alternate space	1	F3
LDUHA	Load unsigned halfword from Alternate space	1	F3
LDA	Load word from Alternate space	1	F3
LDDA	Load doubleword from Alternate space	2	F3
STB	Store byte	2	F3
STH	Store halfword	2	F3
ST	Store word	2	F3
STD	Store doubleword	3	F3
STBA	Store byte to Alternate space	2	F3
STHA	Store halfword to Alternate space	2	F3
STA	Store word to Alternate space	2	F3
STDA	Store doubleword to Alternate space	3	F3
LDF	Load floating point register	1	F3
LDDF	Load double floating point register	1	F3
LDFSR	Load floating point state register	1	F3
LDC	Load coprocessor	1	F3
LDDC	Load double coprocessor	1	F3
LDCSR	Load coprocessor state register	1	F3
STF	Store floating point register	1	F3
STDF	Store double floating point register	1	F3
STFSR	Store floating point state register	1	F3
STDFQ	Store double floating point queue	3	F3
STC	Store coprocessor register	1	F3
STDC	Store double coprocessor register	1	F3
STCSR	Store coprocessor state register	1	F3
STDCQ	Store double coprocessor queue	3	F3
LDSTUB	Atomic load-store unsigned byte	2	F3
LDSTUBA	Atomic load-store unsigned byte in Alternate space	2	F3
SWAP	Atomic swap register with memory	2	F3
SWAPA	Atomic swap register with memory in Alternate space	2	F3
ADD	Add	1	F3
ADDcc	Add and update condition code	1	F3
ADDX	Add with carry	1	F3
ADDXcc	Add with carry and update condition code	1	F3
TADDcc	Tagged add (updates condition code)	1	F3
TADDccTV	Tagged add and trap on overflow	1	F3
SUB	Subtract	1	F3
SUBcc	Subtract and update condition code	1	F3
SUBX	Subtract with borrow	1	F3
SUBXcc	Subtract with borrow and update condition code	1	F3

Table 1 — Instruction Set (Cont)

Mnemonic	Function	Cycles	Format
TSUBcc	Tagged subtract	1	F3
TSUBccTV	Tagged subtract and trap on overflow	1	F3
MULScc	Multiply step and update condition code	1	F3
AND	Logical AND	1	F3
ANDcc	Logical AND and update condition code	1	F3
ANDN	Logical NAND	1	F3
ANDNcc	Logical NAND and update condition code	1	F3
OR	Logical OR	1	F3
ORcc	Logical OR and update condition code	1	F3
ORN	Logical NOR	1	F3
ORNcc	Logical NOR and update condition code	1	F3
XOR	Logical XOR	1	F3
XORcc	Logical XOR and update condition code	1	F3
XNOR	Logical XNOR	1	F3
XNORcc	Logical XNOR and update condition code	1	F3
SETHI	Set most significant 22 bits of register	1	F2
SLL	Logical shift left	1	F3
SRL	Logical shift right	1	F3
SRA	Arithmetic right shift	1	F3
SAVE	Save caller's window	1	F3
RESTORE	Restore caller's window	1	F3
Bicc	Branch on integer condition codes	1	F2
FBfcc	Branch on floating point condition codes	1	F2
CBccc	Branch on coprocessor condition codes	1	F2
CALL	Call subroutine	1	F1
JMPL	Jump and link	2	F3
RETT	Return from trap	2	F3
Ticc	Trap on integer condition codes — untaken	1	F3
	— taken	5	
RDY	Read Y register	1	F3
RDPSR	Read processor state register	1	F3
RDWIM	Read window invalid mask	1	F3
RDTBR	Read trap base register	1	F3
WRY	Write Y register	1	F3
WRPSR	Write processor state register	1	F3
WRWIM	Write window invalid mask	1	F3
WRTBR	Write trap base register	1	F3
UNIMP	Unimplemented instruction		F2
IFLUSH	Flush instruction cache		F3
FPop	Floating point operate:		F3
	FiTO(s,d,x), F(s,d,x)TOi, F(s,d,x)TOiR, FsTOd, FsTOx, FdTOs, FdTOx, FxTOs, FxTOd, FMOVs, FNEGs, FABSs, FSQRT(s,d,x), FADD(s,d,x), FSUB(s,d,x), FMUL(s,d,x), FDIV(s,d,x), FCMP(s,d,x), FCMPE(s,d,x)		
COop	Coprocessor operate		F3

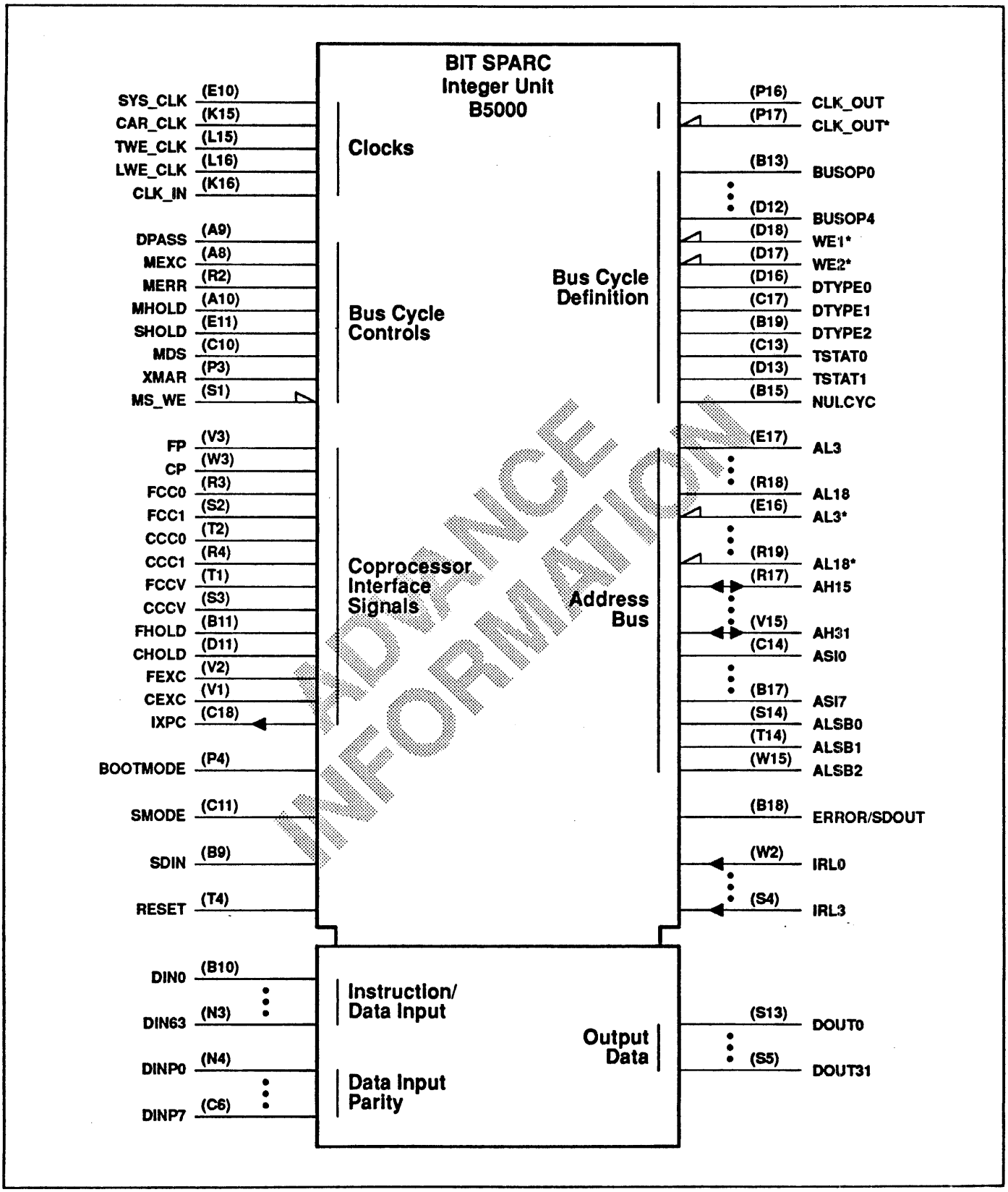


Figure 8 — BIT SPARC Integer Unit Pin Assignments

Table 2 — Signal Summary

Description	Signal Name	Type
Clocks		
IU Main clock	SYS_CLK	Input
Cache address clock	CAR_CLK	Input
Write enable clocks	TWE_CLK, LWE_CLK	Input
System clock input	CLK_IN	Input
Differential output clock	CLK_OUT, CLK_OUT*	Output — 50 Ω
Address Bus		
Address Low	AL [18..3]	Output — 30 Ω
Complementary Address Low	AL [18..3]*	Output — 30 Ω
Address High	AH [31..15]	I/O — 100 Ω
Address Space Identifier	ASI [7..0]	Output — 50 Ω
Byte select	ALSB [2..0]	Output — 50 Ω
Data Bus		
Output data	DOUT [31..0]	Output — 50 Ω
Instruction/data input	DIN [63..0]	Input
Data input parity	DINP [7..0]	Input
Bus Cycle Definition		
Bus operation type	BUSOP [4..0]	Output — 50 Ω
Write enable	WE1*, WE2*	Output — 30 Ω
Data type indicator	DTYPE [2..0]	Output — 50 Ω
Trap status	TSTAT [1..0]	Output — 50 Ω
NULL cycle indicator	NULCYC	Output — 50 Ω
Bus Cycle Controls		
Data pass cycle control	DPASS	Input
Memory exception	MEXC	Input
Memory error	MERR	Input
Hold input control	MHOLD, SHOLD	Input
Memory strobe	MDS	Input
External cache address enable	XMAR	Input
Externally generated write enable	MS_WE	Input
Coprocessor Interface Signals		
Coprocessor present	FP, CP	Input
Condition codes	FCC [1..0], CCC [1..0]	Input
CC bits valid	FCCV, CCCV	Input
Coprocessor hold	FHOLD, CHOLD	Input
Exception	FEXC, CEXC	Input
Increment external PC	IXPC	Output — 50 Ω
Miscellaneous I/O signals		
Interrupt request level	IRL [3..0]	Input
Reset	RESET	Input
Internal error/Scan data out	ERROR/SDOUT	Output — 50 Ω
Enable boot mode	BOOTMODE	Input
Scan mode enable	SMODE	Input
Scan data in	SDIN	Input

Signal Description

Clocks

SYS_CLK — Main Integer Unit clock. All internal timings are based on SYS_CLK.

CAR_CLK — Cache Address Register clock. AL [18..3] and AL [18..3]* are internally clocked by this input. It is expected that CAR_CLK will be early with respect to SYS_CLK for matching wire delay in the cache address path.

LWE_CLK — Leading Edge Write Enable clock allows the system designer to tailor the leading edge of the WE1* and WE2* write signals. LWE_CLK is specified with a set-up time to SYS_CLK, and should be delayed with respect to CAR_CLK to ensure address set-up to WE1*, WE2*.

TWE_CLK — Trailing Edge Write Enable clock determines the placement of the trailing edge of WE1* and WE2*. TWE_CLK is specified with a set-up time to SYS_CLK, and should be advanced with respect to CAR_CLK to ensure address hold to WE1*, WE2*.

CLK_IN — This clock input is only used to generate CLK_OUT. No internal timing constraints are placed on this clock, but it is expected that it will be early with respect to CAR_CLK.

CLK_OUT, CLK_OUT* — Clock output from the Integer Unit. It is expected that SYS_CLK will be derived from this clock, as will other system clocks. CLK_OUT* is a differential version of CLK_OUT.

Address Bus

AL [18..3], AL [18..3]* — The address low bus is typically used for addressing the cache. These bits become valid based upon a rising edge of the CAR_CLK input. If the clock is skewed early with respect to the main IU clock, wiring delay for the cache addresses may be matched to achieve maximum performance. AL [18..3]* are complementary versions of AL [18..3].

AH [31..15] — Address high bus provides the most significant 17 bits of address. Typically used for cache tag compare and virtual page mapping, this bus is timed such that it must be latched externally with SYS_CLK; AH [31..15] become valid in the E stage during loads and stores. Additionally, these pins are bidirectional with an internal feedback path to AL[18..2] to allow a cache controller access to the cache data and tags during MHOLD cycles, at which time these pins are configured for input.

ASI [7..0] — The Address Space Identifier bus describes one of 256 different address spaces. Used in conjunction with the 32 bit address bus, ASI allows addressing of up to 1 Tera byte of virtual memory. These pins are valid during the W stage of every memory access instruction.

ALSB [2..0] — The three least significant address bits indicate which byte, halfword, or word is selected for the current memory access.

Data Bus

DOUT [31..0] — Store data as well as coprocessor instructions and addresses are output on this bus. Store data is valid during the W stage of memory store cycles. Coprocessor (including floating point) instructions are valid during the D stage of related coprocessor cycles. When this bus is not being used to transfer data or CP, FP instructions/addresses, the current D stage instruction address is output.

DIN [63..0] — Instructions and data are loaded through this bus. Since it is two 32-bit words wide, the processor can fetch two instructions simultaneously. This is especially useful for branch instructions since the branch target can be fetched while still keeping the next sequential instruction internally buffered, thus allowing single cycle branches. The IU uses this bus to fetch two instructions, or load one data word. The timing of this bus is with respect to SYS_CLK.

DINP [7..0] — All input data bytes are associated with an ODD parity indicator bit. These parity bits are checked every cycle, and generate a data parity error trap when enabled.

Bus Cycle Definition

BUSOP [4..0] This five bit bus is used to indicate the type of external operation to be performed during the next write stage. Table 3 on the following page shows the different codes, with a short description of each.

WE1*, WE2* Write enable signals to the cache are controlled by the IU to minimize skew with respect to the addresses. Two clocks, LWE_CLK and TWE_CLK, are used to place the edges of the write enable pulse, providing flexibility for the system designer using different manufacturers memory devices. Two separate outputs are provided to minimize loading and wiring delay on either one. These pins are identical both in functionality and timing.

DTYPE [2..0] Each processor cycle, the DTYPE bus describes which type of data is currently being driven on the DOUT [31..0] bus. The following table describes the options.

DTYPE	Type of Data
0	FPU Instruction
1	Store Data
2	CP Instruction
3	Decode stage PC
4	Previous FPU Instruction
5	reserved
6	Previous CP Instruction
7	NULL (MHOLD)

TSTAT [1..0] The current status of processor traps is indicated with this bus. The four options are described in the following table.

TSTAT	Trap Type
0	No trap
1	FP-Trap
2	CP-Trap
3	Trap

NULCYC Output indicating that the current bus access should be cancelled. Only used during instruction fetch cycles, NULCYC is equivalent to a NULL BUSOP code in the previous cycle.

Bus Cycle Controls

DPASS During MHOLD, an external controller may pass data directly from the D_{in} input of the IU to the D_{out} output by asserting DPASS. The state of AH 15 determines which half of the 64-bit word will be placed on the output; when AH 15 is high the most significant word is passed, and when AH 15 is low the least significant word is selected.

MEXC Memory exceptions such as page fault from a Memory Management Unit are indicated using this pin. MEXC informs the processor that the requested instruction or data is not present at the data input of the IU. Asserted in lieu of valid data during the W stage of the current access cycle or by using MDS during MHOLD, this signal informs the processor of an exception from which the cache controller or MMU cannot recover.

MERR This pin is typically used to signal a hard error discovered during a memory access. Parity errors or ECC exceptions are two examples.

MHOLD The Memory HOLD input is used to force the processor into a hold state in the next cycle. Typically used to inform the IU of a cache miss, this signal causes the processor to place the AH [31..15] I/O signals in input mode to allow a cache controller access to the cache address bus. MHOLD may be asserted for any number of cycles, but the IU sustains the hold for 2 cycles after it is released.

SHOLD External system devices may force the IU into a hold state by asserting this pin.

MDS Memory Data Strobe; In a cache based system, MHOLD will be used to indicate a cache miss. Since MHOLD is not asserted until after the completion of the cycle, the IU will have strobed invalid data into its instruction register. This signal enables the cache controller to strobe valid instructions or data into the IU during an MHOLD cycle.

XMAR During miss processing in a cache based system, the cache controller determines the source of AL [18..5] using this signal. When asserted, the address pins are derived from inputs AH [31..18]. When low, the IU generates AL [18..5] with a copy of the miss address. In either case, the cache controller provides AL [4..3], ALSB [2] using pins AH [17..15].

MS_WE Cache write enable signals are activated during an MHOLD cycle when this signal is asserted. An external cache controller might use this pin to enable writing data and tags to the cache during cache misses. Timing of the WE1* and WE2* pins is controlled by the write enable clocks.

Table 3 — BUSOP Codes

BUSOP [4..0]	Description
Special Codes	
00000	NULL — indicates no bus operation
00001	ldstub — this code is active to indicate the initiation of the atomic instruction LDSTUB
00010	I-Fetch — indicates the processor is fetching a sequential instruction from memory
00011	swap — this code is active to indicate the initiation of the atomic instruction SWAP
00100	Hldd — signifies the second cycle of a two cycle load instruction such as LDD
00101	Hstd — signifies the second cycle of a four cycle store instruction such as STD
00110	Hst1 — active in the next to last cycle of a store access
00111	Hst2 — active during the last cycle of a store access
Integer Codes	
01000	ldh — signifies a load data halfword instruction cycle
01001	ldb — signifies a load data byte instruction cycle
01010	ldd — signifies the first cycle of a load double data word instruction
01011	ld — signifies a load data word instruction cycle
01100	sth — signifies the first cycle of a store data halfword
01101	stb — signifies the first cycle of a store data byte
01110	std — signifies the first cycle of a store double data word
01111	st — signifies the first cycle of a store data word
Floating Point	
10000	ldf — active during the cycle of a load floating point operand
10001	ldfsr — indicates a load floating point state register instruction
10010	*reserved*
10011	lddf — indicates a load double floating point operand
10100	stf — indicates a store floating point operand instruction
10101	stfsr — signifies store floating point state register
10110	stdfq — signifies the store floating point queue instruction
10111	stdf — indicates the store double floating point operand instruction
Coprocessor	
11000	ldc — active during the cycle of a load coprocessor operand
11001	ldcsr — indicates a load coprocessor state register instruction
11010	*reserved*
11011	lddc — indicates a load double coprocessor operand
11100	stc — indicates a store coprocessor operand instruction
11101	stcsr — signifies store coprocessor state register
11110	stdcq — signifies the store coprocessor queue instruction
11111	stdc — indicates the store double coprocessor operand instruction

FP Interface

FP — A high level on FP indicates to the IU that a floating point coprocessor is present.

FCC [1..0] — Four floating point coprocessor condition codes are interpreted by the IU according to the following table:

FCC [1..0]	Interpretation
0	equal (=)
1	less than (<)
2	greater than (>)
3	unordered (?)

FCCV — The floating point condition codes are defined as valid when this signal is active.

FHOLD — FHOLD performs similar to SHOLD; it (FHOLD) is reserved for use by the floating point coprocessor.

FEXC — The floating point coprocessor signals an exception by activating this pin. The IU will respond by taking a floating point Trap, signalled on the TSTAT bus. In order to guarantee that the IU recognizes the trap, this signal must be held active until FP-Trap is asserted.

IXPC — When the IU asserts this signal, the coprocessors should increment their copies of the Program Counter by four bytes.

CP Interface

CP — A high level on CP indicates to the IU that a user defined coprocessor is present.

CCC [1..0] — Four coprocessor condition codes, similar to the floating point condition codes, are interpreted by the IU.

CCCV — The coprocessor condition codes are defined as valid when this signal is active.

CHOLD — CHOLD performs exactly as SHOLD; it (CHOLD) is reserved for use by the coprocessor.

CEXC — The user defined coprocessor signals an exception by activating this pin. The IU will respond by taking a coprocessor Trap, signalled on the TSTAT bus. In order to guarantee that the IU recognizes the trap, this signal must be held active until CP-Trap is asserted.

Miscellaneous I/O

IRL [3..0] — Sixteen levels of interrupt are defined by the SPARC architecture, encoded on the 4-bit IRL field. These interrupts are asynchronous, and the priority is from 0 to 15 with 0 the lowest. All interrupts but 15 may be masked by the PIL field in the PSR register.

RESET — A reset may be applied to the processor at any time, causing all internal registers to be initialized and execution to begin at address zero. RESET must be asserted for at least 10 SYS_CLK cycles.

BOOTMODE — This signal places the processor in boot mode for the internal register file and allows access to the Bootmode Configuration Register. Description of this boot mode may be found in a later section.

SMODE — Diagnostics may be performed on the IU through the use of scan mode, controlled with this input. When asserted, clocking of non-scanned registers is disabled and scanned registers are internally connected as one long serial shift register. This register is clocked with SYS_CLK.

SDIN — A serial stream of data is input through this pin when the processor is connected in scan mode.

ERROR/ SDOUT — This pin has two meanings depending on the state of SMODE. In run mode (SMODE = "0"), if the IU encounters an irrecoverable error, it enters a HALT state and asserts this signal. The only way to recover is to have RESET asserted. In scan mode (SMODE = "1"), serial data is shifted out on this pin.

Electrical Specifications

Table 4 — Absolute Maximum Ratings

Parameter	Symbol	Value	Units
Supply Voltage	V_{ee}	–7.0 to 0	V_{dc}
Input Voltage	V_{in}	V_{ee} to 0	V_{dc}
Output Source Current			
Continuous (50 Ω output)	I_{oc}	30	mA_{dc}
Continuous (30 Ω output)	I_{oc}	50	mA_{dc}
Surge	I_{os}	100	mA_{dc}
Storage Temperature	T_{st}	–55 to 150	°C
Junction Temperature	T_j	165	°C

Note: Permanent damage may occur if any one absolute maximum rating is exceeded. Functional operation is not implied and device reliability may be impaired by exposure to higher than recommended conditions.

Table 5 — Recommended Operating and Test Conditions

Parameter	Symbol	Min	Nom	Max	Units
Supply Voltage (GND = 0)	V_{ee}	–5.46	–5.20	–4.94	V_{dc}
Ambient Temperature	T_a *	0		70	°C
Junction Temperature	T_{jt}			125	°C
Output Termination to –2.0 V_{dc}	R_t **		50		Ω

* 500 to 1000 Linear Feet per Minute Ambient Airflow.

** AL [18..3], AL [18..3]*, WE2*, and WE1* use 30 Ω output termination.

Table 6 — DC Characteristics

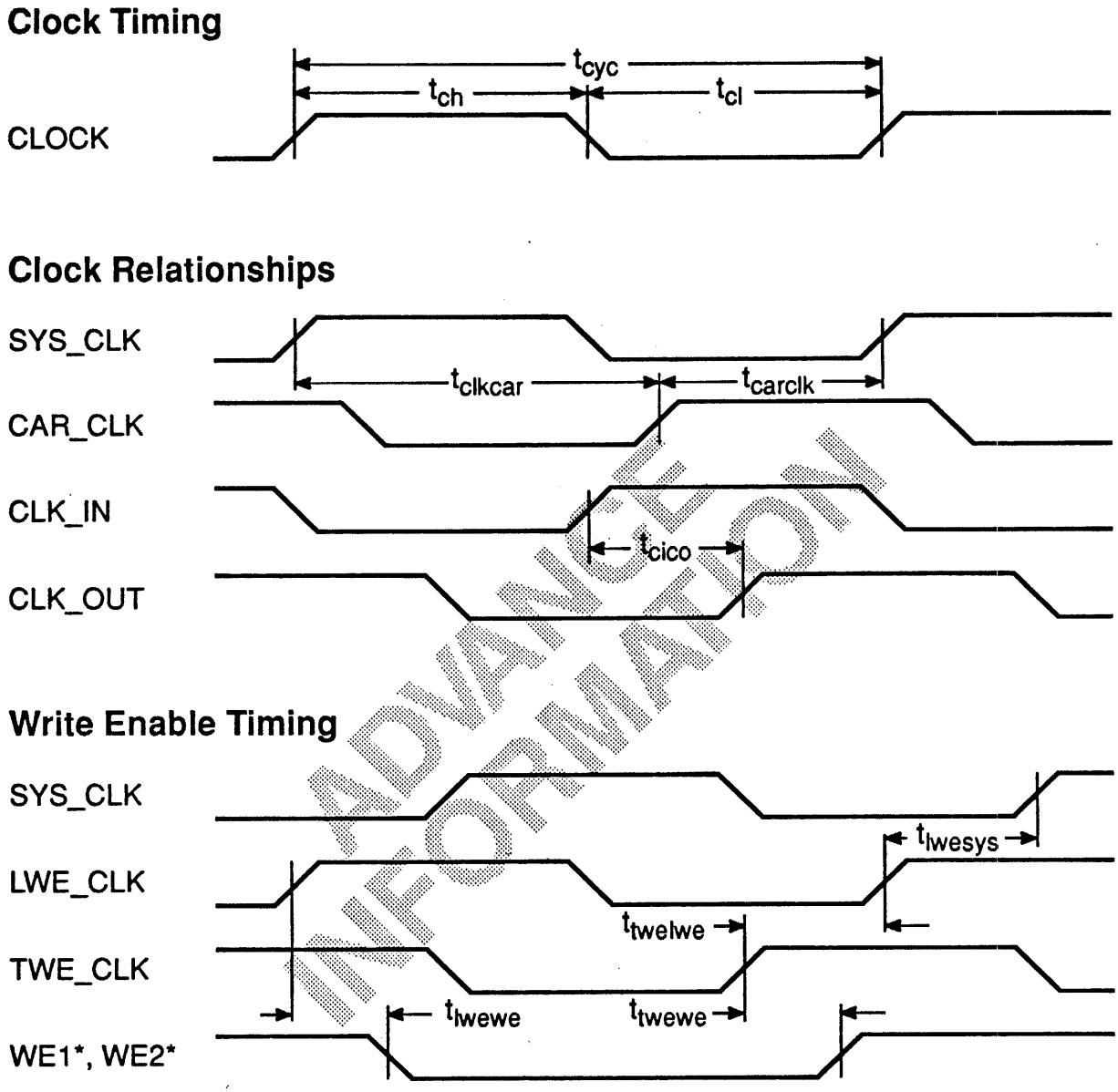
Parameter	Symbol	0 °C		25 °C		70 °C		Units
		Min	Max	Min	Max	Min	Max	
Input Voltage High	V_{ih}	–1.17	–0.84	–1.13	–0.81	–1.07	–0.735	V_{dc}
Input Voltage Low	V_{il}	–1.95	–1.48	–1.95	–1.48	–1.95	–1.45	V_{dc}
Output Voltage High	V_{oh}	–1.02	–0.84	–0.98	–0.81	–0.92	–0.735	V_{dc}
Output Voltage Low	V_{ol}	–1.95	–1.63	–1.95	–1.63	–1.95	–1.60	V_{dc}

Table 7 — DC Characteristics

Parameter	Symbol	Min	Nom	Max	Units
Supply Current	I_{ee}				A_{dc}
Input Current High	I_{ih}			0.3	mA_{dc}
Input Capacitance	C_{in}		5		pF

Table 8 — Switching Characteristics

Parameter	Symbol	Min	Max	Units
Clock Timing				
Cycle Time	t_{cyc}	12.5		ns
Clock Pulse Width High	t_{ch}	5.5		ns
Clock Pulse Width Low	t_{cl}	5.5		ns
LWE_CLK to SYS_CLK Set up	t_{lwesys}			ns
TWE_CLK to LWE_CLK Time	t_{twelwe}			ns
LWE_CLK to Write Enable Low	t_{lwewe}			ns
TWE_CLK to Write Enable High	t_{twewe}			ns
CLK_IN to CLK_OUT Delay	t_{cico}			ns
CAR_CLK Delay from SYS_CLK	t_{clkcar}			ns
CAR_CLK to SYS_CLK Set up	t_{carclk}			ns
Setup and Hold Timing				
Setup time to SYS_CLK	t_{su}			ns
Hold time from SYS_CLK	t_{hld}			ns
Clock Timing				
CAR_CLK to AL [18..3], AL [18..3]*	t_{ccal}			ns
SYS_CLK High to AH [31..15]	t_{chah}			ns
SYS_CLK Low to AH [31..15]	t_{clah}			ns
SYS_CLK to ALSB[2..0]	t_{calsb}			ns
SYS_CLK to ASI	t_{casi}			ns
SYS_CLK to BUSOP [4..0]	t_{cbop}			ns
SYS_CLK to TSTAT [1..0]	t_{ctst}			ns
SYS_CLK to NULCYC	t_{cnul}			ns
SYS_CLK to DTYPE [2..0]	t_{cdtyp}			ns
SYS_CLK to IXPC	t_{cixpc}			ns
SYS_CLK to ERROR	t_{cerr}			ns
SYS_CLK High to DOUT [31..0]	t_{cdout}			ns
Mhold Timing				
MHOLD Setup Time	t_{msu}			ns
MHOLD Hold Time	t_{mhld}			ns
Clocked MHOLD to AH [31..15] Inactive	t_{mahi}			ns
MHOLD to CAR_CLK HIGH	t_{mcar}			ns



CLOCK can be any of the following: SYS_CLK, CLK_IN, CAR_CLK, LWE_CLK, and TWE_CLK

***Figure 9** — Timing Diagrams*

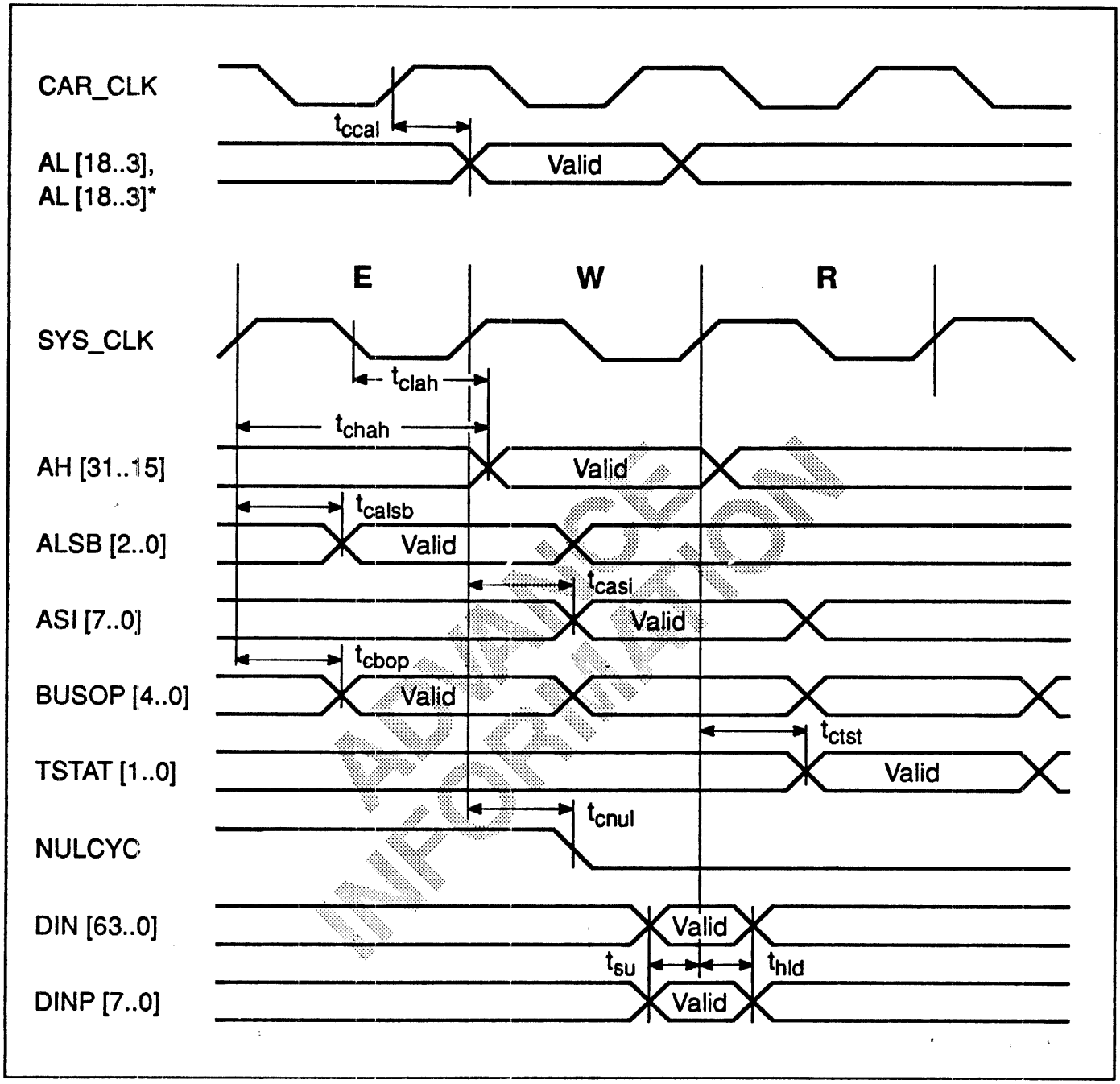


Figure 10 — Load/Fetch Timing with No MHOLD

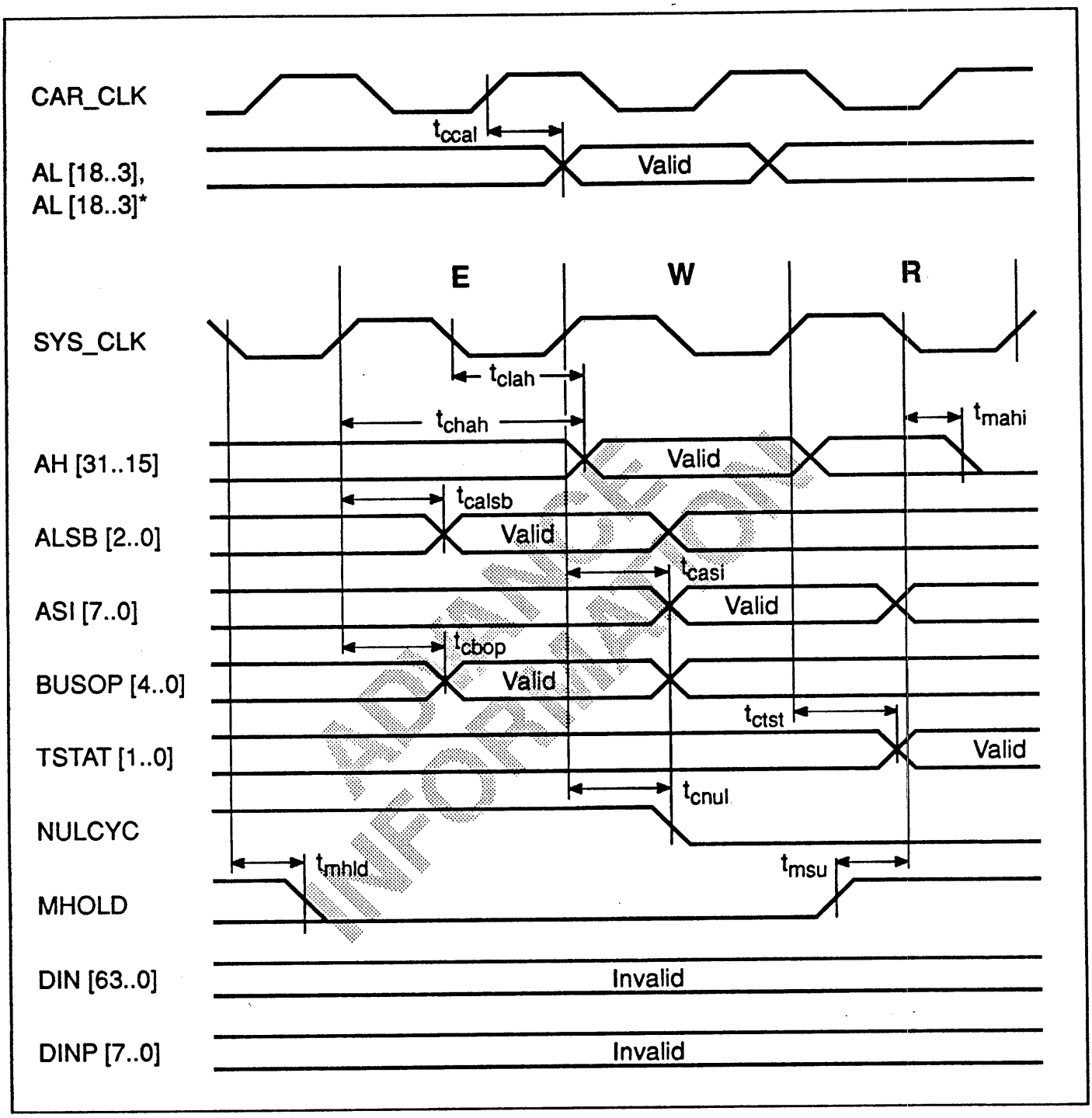


Figure 11 — Load/Fetch Timing with MHOLD

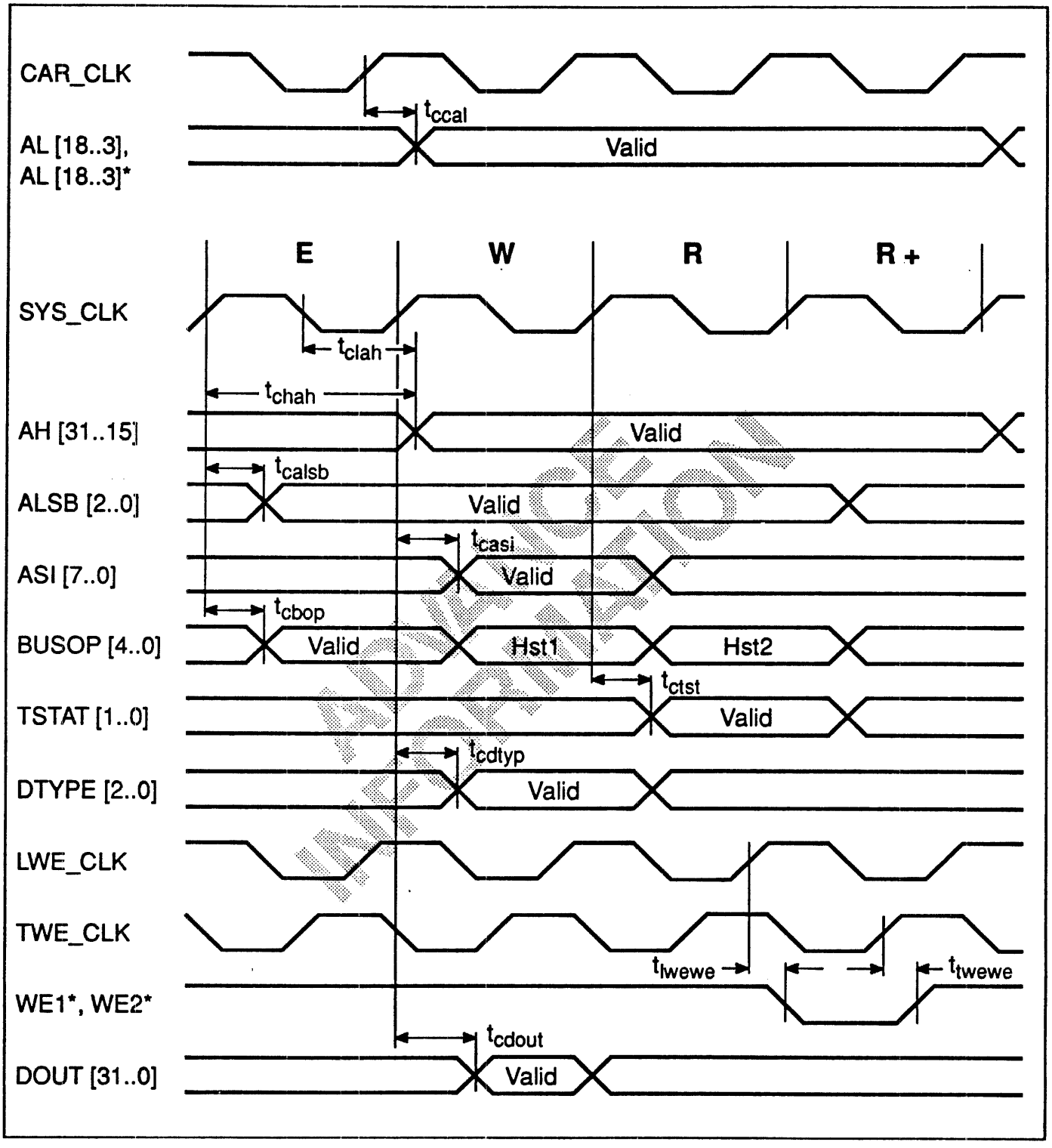


Figure 12 — Store Timing with No MHOLD

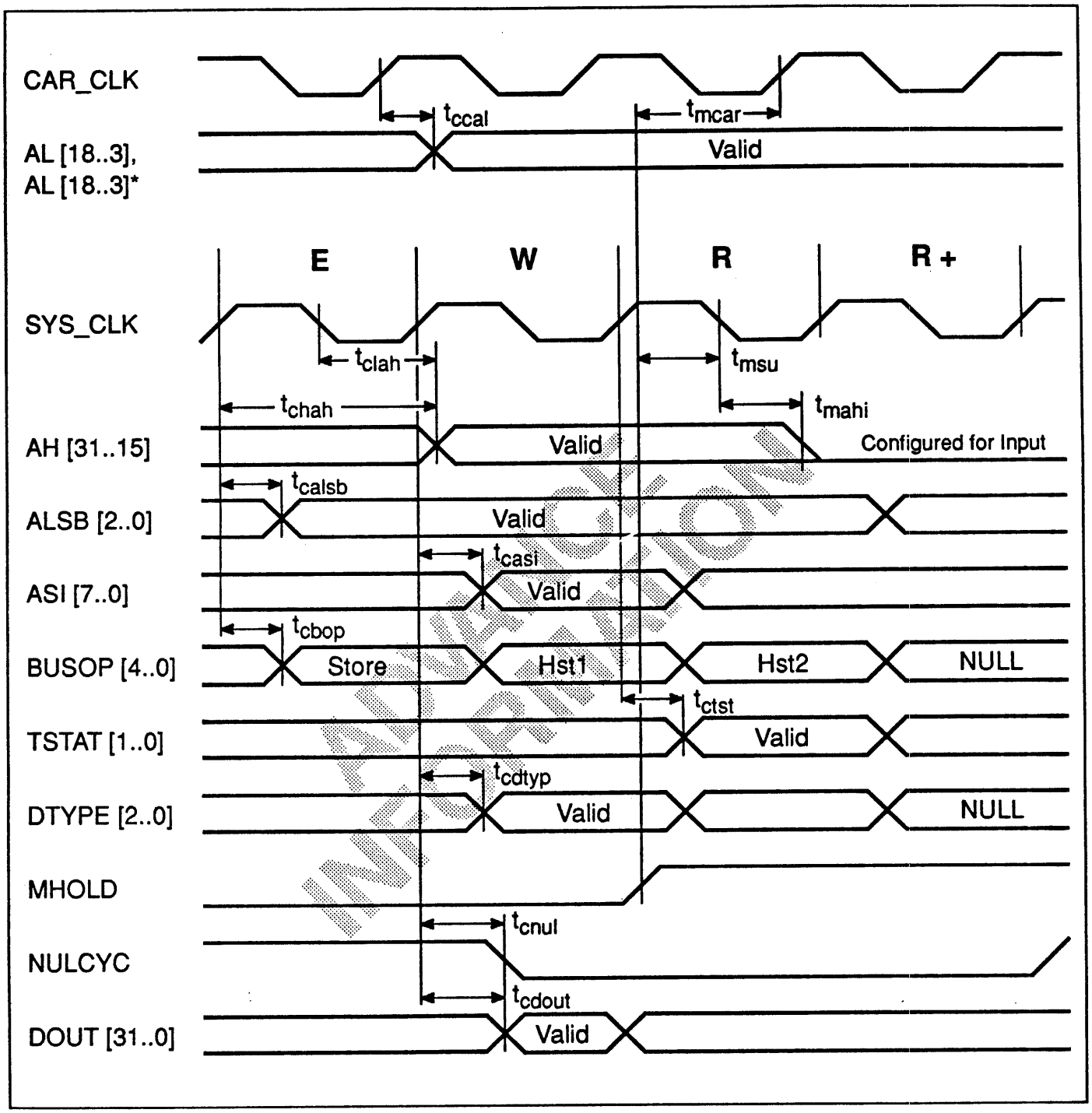


Figure 13 — Store Timing with MHOLD

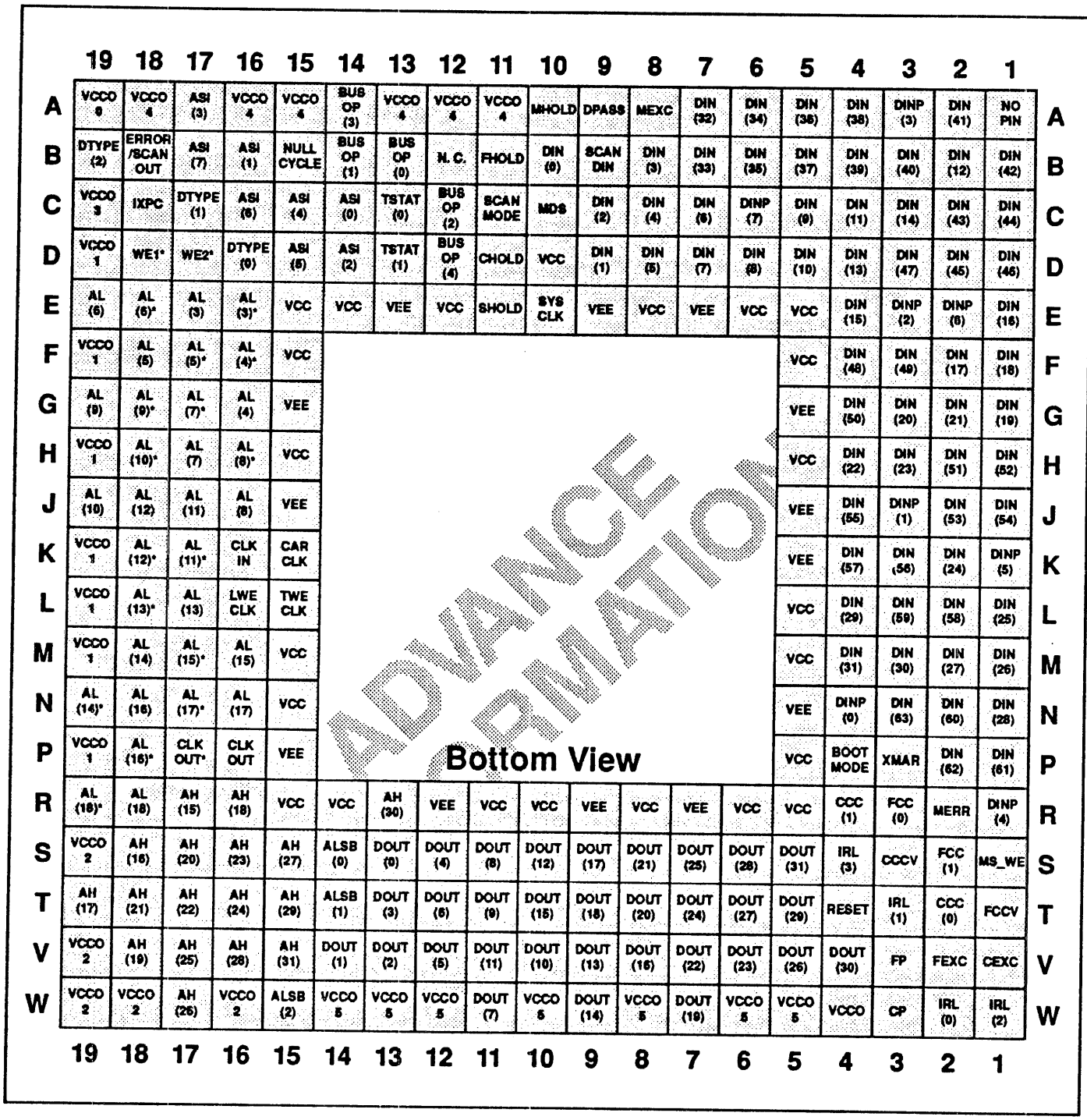


Figure 14 — B5000 Pin Out

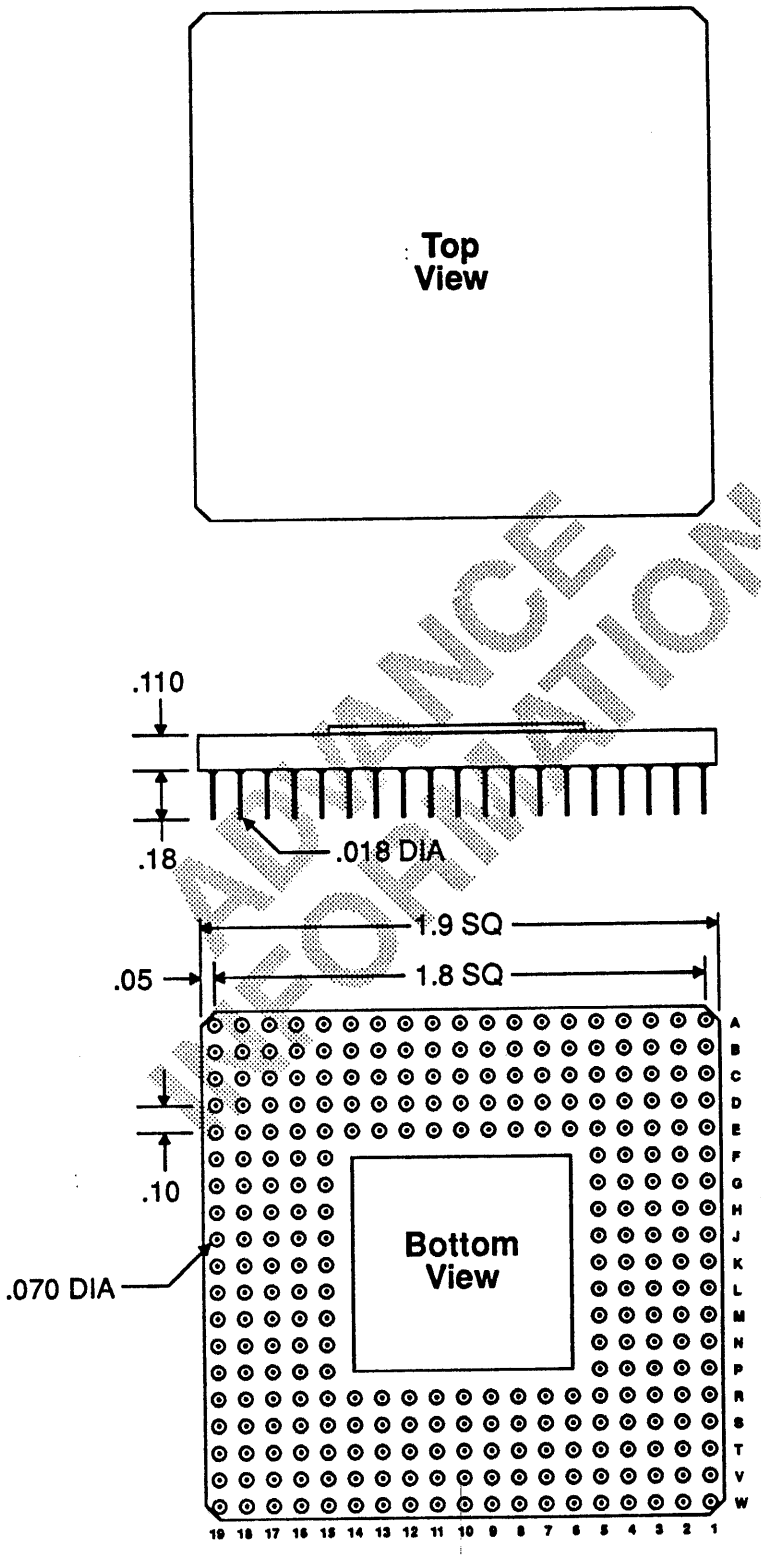


Figure 15 — B5000 Package Dimensions

BIT Bipolar Integrated Technology, Inc.

Western Area Sales Office:
12930D Saratoga Ave.
Saratoga, CA 95070
(408) 996-2060

Eastern Area Sales Office:
5680 Peachtree Parkway, Suite E
Norcross, GA 30092
(404) 446-7353

Southwestern Regional Sales Office:
575 Anton Blvd., 3rd Floor
Costa Mesa, CA 92626
(714) 432-6396

BIT SALES REPRESENTATIVES

Alabama	Eastern Area Sales Office (404) 446-7353
Alaska	Western Area Sales Office (408) 996-2060
Arizona	Compass Marketing (602) 996-0635
Arkansas	Nova Marketing (214) 750-6082
California	
Northern	Brooks Technical Group (415) 960-3880
Southern	SW Regional Sales Office (714) 432-6396
San Diego Co.	Littlefield & Smith (619) 455-0055
Colorado	Western Area Sales Office (408) 996-2060
Connecticut	Coakley, Boyd & Abbett (203) 239-6217
Delaware	Eastern Area Sales Office (404) 446-7353
District of Columbia	Eastern Area Sales Office (404) 446-7353
Florida	Eastern Area Sales Office (404) 446-7353
Georgia	Eastern Area Sales Office (404) 446-7353
Hawaii	Brooks Technical Group (415) 960-3880
Idaho	ES/Chase (503) 292-8840
Illinois	Phase II Marketing (312) 806-1330
Indiana	Tritech Sales (313) 553-3370
Iowa	Cahill, Schmitz & Howe (319) 377-8219
Kansas	Western Area Sales Office (408) 996-2060
Kentucky	
Eastern	Hester Associates (513) 866-6699
Western	Tritech Sales (313) 553-3370
Louisiana	Nova Marketing (214) 750-6082
Maine	Coakley, Boyd & Abbett (617) 444-5470
Maryland	Eastern Area Sales Office (404) 446-7353
Massachusetts	Coakley, Boyd & Abbett (617) 444-5470
Michigan	Tritech Sales (313) 553-3370 (616) 382-5570
Minnesota	Cahill, Schmitz & Cahill (612) 646-7217
Mississippi	Eastern Area Sales Office (404) 446-7353
Missouri	Western Area Sales Office (408) 996-2060
Montana	ES/Chase (503) 292-8840
Nebraska	Western Area Sales Office (408) 996-2060
Nevada	Brooks Technical Group (415) 960-3880
Clark Co. only	SW Regional Sales Office (714) 432-6396
New Hampshire	Coakley, Boyd & Abbett (617) 444-5470
New Jersey	
Northern	ERA Inc. (800) 645-5500
Southern	Eastern Area Sales Office (404) 446-7353
New Mexico	Compass Marketing (505) 888-0800
New York	
Westchester Co. & Long Island	ERA Inc. (516) 543-0510
Upstate NY	Alan Lupton Associates (716) 381-4800
North Carolina	Eastern Area Sales Office (404) 446-7353
North Dakota	Cahill, Schmitz & Cahill (612) 646-7217
Ohio	Hester Associates (216) 247-6655 (513) 866-6699
Oklahoma	Nova Marketing (918) 660-5105
Oregon	ES/Chase (503) 292-8840
Pennsylvania	
Eastern	Eastern Area Sales Office (404) 446-7353
Western	Hester Associates (412) 285-1313
Rhode Island	Coakley, Boyd & Abbett (617) 444-5470
South Carolina	Eastern Area Sales Office (404) 446-7353
South Dakota	Cahill, Schmitz & Cahill (612) 646-7217
Tennessee	Eastern Area Sales Office (404) 446-7353
Texas	Nova Marketing (214) 750-6082
El Paso Co.	Compass Marketing (602) 996-0635
Utah	Harris/CSI Inc. (801) 467-2299
Vermont	Coakley, Boyd & Abbet (617) 444-5470
Virginia	Eastern Area Sales Office (404) 446-7353
Washington	ES/Chase (206) 823-9535
West Virginia	Hester Associates (513) 866-6699 (412) 285-1313
Wisconsin	
Western	Cahill, Schmitz & Cahill (612) 646-7217
Eastern	Phase II Marketing (414) 272-1401
Wyoming	Western Area Sales Office (408) 996-2060
Canada	Electronic Sales Professionals (613) 236-1221 (416) 626-8221 (514) 691-8129
B.C. only	Western Area Sales Office (408) 996-2060
Israel	Tritech Ltd. (972) 3-498802 or (972) 3-497816

1050 N.W. COMPTON DRIVE
Beaverton, OR 97006
(503) 629-5490

MKTG-D010