Bipolar Integrated Technology, Inc.

Advance Information

B5100

BIT SPARC Floating Point Controller

Features

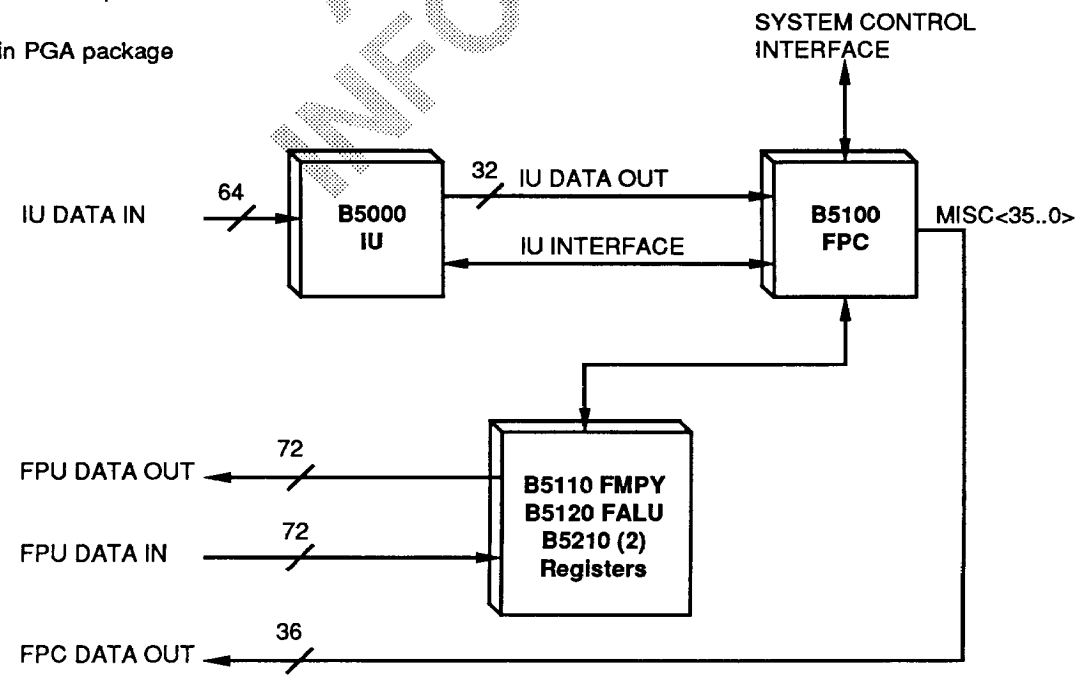
- Fully compatible with the SPARC coprocessor interface definition
- Supports high performance floating point calculations using the BIT B5110/B5120
- Contains an on-chip 4 x 64-bit internal floating point instruction queue
- Supports a 32 x 36-bit floating point register bank by providing controls to two B5210 register files
- Operates concurrently with the BIT SPARC IU
- Supports separate floating point load/store memory access paths
- Utilizes the parity generation and checking capability of the B5110 and B5120
- FPC generates byte parity on the peripheral access bus
- 80 MHz operation
- ECL 10KH compatible interface
- 259 pin PGA package

Description

The BIT SPARC™ Floating Point Controller is an ECL VLSI support circuit for the BIT SPARC family of microprocessors and peripherals. Implemented in a 1.2 micron ECL technology, the FPC provides interface to, and control of, a high performance floating point processing subsystem consisting of the BIT floating point chip set (B5110/B5120) and two register files (B5210's).

Maximum data throughput for the floating point subsystem is achieved using separate load and store buses. These data paths are 72-bits wide, enhancing double precision load/store performance. Concurrent operation of the Integer Unit (IU) and Floating Point Controller (FPC) further increases overall system performance. The IU dispatches floating point operations to the FPC and continues executing instructions; the FPC will store these instructions in an on-chip instruction queue and execute them when ready.

All register addresses and floating point opcodes are generated using internal state machines. Instructions are executed using concurrent ALU and multiplier operation in a register to register fashion.



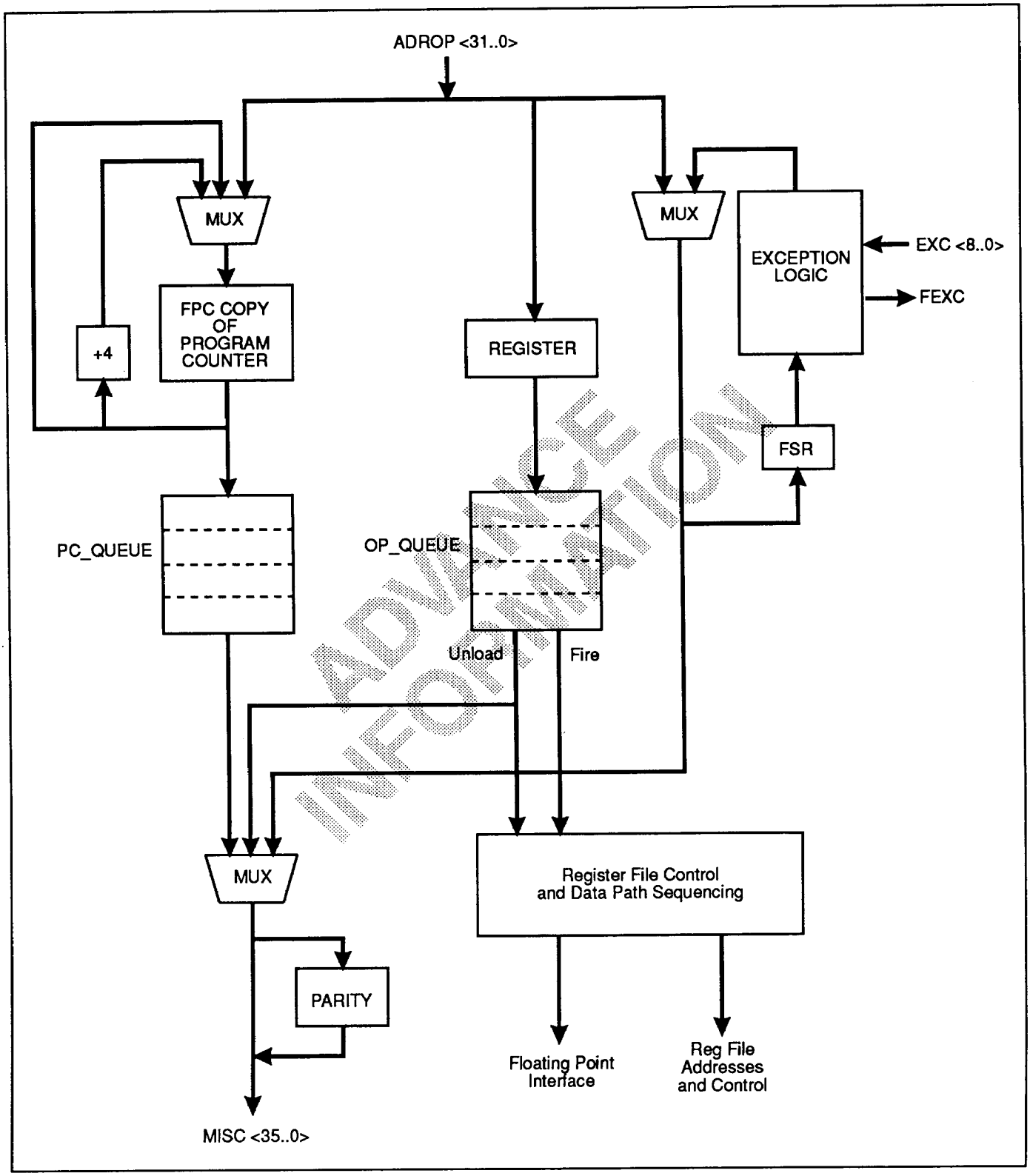


***Figure 1** — BIT SPARC Floating Point Controller Block Diagram*

Functional Description

FPC Overview

The B5100 floating point controller is a VLSI support circuit for BIT's ECL implementation of the SPARC architecture. It provides control, addresses, and op-codes to the B5110/ B5120 (FMPY/FALU) floating point chip set and two B5210 register files. These standard BIT products form an IEEE compatible floating point arithmetic unit (FAU or FPU) for this BIT SPARC implementation.

Fully compatible with the SPARC architectural description of a floating point coprocessor, the FPC recognizes floating point instructions sent from the IU, places them into a queue, and processes them when the desired register operands are available. Once the IU has dispatched the floating point instruction to the FPC, it is free to continue executing integer instructions concurrently with the FPU.

Additionally, because the FPC is capable of concurrent execution between the FMPY and the FALU, it achieves the highest possible performance from a given instruction stream.

Important Features

SPARC compatibility - the FPC fully supports the SPARC coprocessor architecture.

IEEE compatibility - a SPARC floating point subsystem built with the FPC and BIT standard products maintains compatibility with the ANSI/IEEE 754-1985 standard for floating point computation.

Register to register architecture - all floating point operations are performed on register operands and written back into the register file. Only floating point load and store instructions access memory.

Concurrent operation - the FPU executes concurrently with the IU; additionally, the FPC operates the FMPY and FALU in a parallel fashion for certain instruction mixes.

Large register file - 32 single precision or 16 double precision registers are controlled and addressed by the FPC. These registers are provided by two B5210 register file devices.

Hardwired control - all instruction dispatching, IU communication, register file addressing, and exception handling are performed with hardwired state machines.

Internal instruction queue - instructions are placed in an internal queue for temporary storage until the desired operands are available. Four places deep, the instruction queue holds both the floating point instruction and its associated program counter address.

Floating Point Interface

The BIT SPARC architecture provides two separate coprocessor ports, a floating point (coprocessor) port and a user-defined coprocessor port. The IU fetches and dispatches floating point/coprocessor instructions from the same stream as the integer instructions. The floating point port is defined by the SPARC architecture and is supported by the SPARC instruction set.

Two classes of floating point instructions are included, FPU loads/stores and FPU operate (Fpop). The IU executes floating point loads and stores by generating the address and control signals required for reading/writing memory. A path is provided from cache or main memory directly to the FPU 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 FPU. The FPU then completes the instruction concurrently with the IU operation.

A functional block diagram of the B5100 floating point controller is shown in Figure 2.

Traps

If the processor determines that it cannot continue the execution of an instruction because of an exceptional condition, it generates a trap. An instruction is therefore defined as trapped if any trap occurs during its execution. Two types of traps are recognized by the BIT SPARC IU; synchronous (internal or software controlled) and asynchronous (external or interrupt controlled). To the IU, FPU traps are always synchronous because they occur at the beginning of an Fpop instruction. The traps are therefore taken before the instruction can cause an undesirable change in system state.

Traps save the program counter and next program counter, either of the instruction which caused the synchronous trap or the instruction that was about to execute when an interrupt or coprocessor trap occurred. Because the FPU operates

independently of the IU, when an fp_trap occurs it will always trap the next fp-instruction processed by the IU. The causative instruction is therefore stored at the top of the floating point queue. The remaining element(s) in the queue will point to Fpops that have been dispatched by the IU, but not completed by the FPU, so that they may be emulated or re-executed.

When the FPU traps an exception, it enters an "exception_pending" state, where it remains until the IU takes the fp_exception trap. (This will occur when the IU encounters the next fp-instruction in the instruction stream.) When the IU takes the trap, the FPU switches to an "exception_mode" state, where it remains until the fp_queue has been emptied by one or more STDFQ instructions. The trap is then cleared and normal operations may resume.

FPU REGISTERS

The Floating Point Unit has three kinds of registers associated with it: 32 32-bit working f-registers, a Floating point State Register (FSR), and a Floating point Queue (FQ).

Working Registers

32 f-registers are provided by two B5210 register file chips. All control and address signals for these devices are provided by the FPC. The B5210's 5-port architecture is fully utilized by BIT SPARC's 72-bit direct memory operand-load path and its 72-bit direct memory store path.

Floating Point State Register

This 32-bit register contains various fields describing mode and status of the FPU. Two instructions, LDFSR and STFSR, access the register directly, whereas fields within the FSR may be modified by various instructions. For example, the floating point condition codes are affected by compare instructions, etc. Figure 3 shows the different fields in the FSR. An explanation of these fields follows.

RD - Rounding Direction; these 2-bits select the rounding mode for floating point results. RD allows systems whose destinations are always double or extended precision to mimic, in the absence of over/underflow, the precision of systems with single and double destinations. e.g., the results of higher precision arithmetic operations could be rounded to a lower precision while being stored in the higher precision format. They are encoded as follows:

RD	Rounding
0	round to nearest, even
1	round to zero
2	round to +infinity
3	round to -infinity

RP - Rounding Precision; in accordance with the IEEE specification, extended precision results may be rounded per the following 2-bit coding:

RP	Rounding Precision
0	Extended (63-bits)
1	Single (23-bits)
2	Double (52-bits)
3	unused

TEM - Trap enable mask; these bits define which floating point exceptions caused an fp_exception trap. If one or more of these bits are set to one, and one or more of the corresponding exceptions occur, the FPC generates an IEEE_exception trap.

TEM BIT	Exception enabled
27	Invalid operation
26	Overflow
25	Underflow
24	Divide by Zero
23	Inexact

RD	RP	TEM	reserved	*ftt*	*qne*	res	*fcc*	*aexc*	*cexc*
31:30	29:28	27..23	22..17	16..14	13	12	11:10	9..5	4..0

***Figure 2** — Floating Point Controller State Register (PSR)*

reserved - These bits are reserved for future use. LDSFR should load zeroes into these bits and zeroes will be read from these bits during STFSR.

ftt - Floating point trap type; these bits identify the type of fp_exception trap that has occurred. This data remains valid until the completion of the next Fpop instruction. Unless otherwise indicated below, this data does not affect other fields in the FSR register, nor does it affect the destination f-register.

ftt	Trap type
0	No trap
1	IEEE_exception
2	Unfinished_Fpop
3	Unimplemented_Fpop
4	Sequence error

When an IEEE_exception is indicated, the exception type will be identified in the cexc field.

Unfinished_Fpop indicates the FPU was unable to generate correct IEEE results/exceptions for the decoded instruction.

Unimplemented_Fpop indicates the FPU decoded an unimplemented instruction.

A sequence_error trap indicates the FPU decoded an Fpop instruction before a previously pending fp_exception was handled by the software.

qne - Queue not empty; used after fp_exception traps or STDFQ instructions to determine if the internal queue is empty. STFSR will read this bit and LDFSR will have no affect on it.

fcc - Floating point condition codes; identifies conditions used by the floating point branch instruction (FBfcc). Set by the floating point compare instructions (FCMP and FCMPE), the fcc bits indicate the relationship between floating point operands. In the following table, Fop1 corresponds to the operand held in the f_register addressed by RS1 in the floating point instruction, and Fop2 corresponds to the f_register addressed by RS2. The question mark (when fcc = 3) indicates an unordered relationship, which is the case if either fop1 or fop2 is a NaN (signalling or quiet).

fcc	Relationship
0	Fpop1 = Fpop2
1	Fpop1 < Fpop2
2	Fpop1 > Fpop2
3	Fpop1 ? Fpop2

aexc - Accrued exception bits; when an exception occurs which is masked from causing a trap, the corresponding bit in the aexc field is set. These bits will remain set until specifically reset by LDFSR. A one indicates an accrued exception and a zero indicates no exception.

aexc bit	Exception accrued
9	Invalid operation
8	Overflow
7	Underflow
6	Divide by Zero
5	Inexact

cexc - Current exception bits; indicate the exception status of the most recently executed floating point instruction. Whenever an IEEE exception occurs, these bits will indicate the type of exception. They can be read by and written to by STFSR and LDFSR respectively, but are updated at the completion of every floating point operation. The state of these bits is undefined following a floating point instruction which traps.

cexc bit	Current exception
4	Invalid operation
3	Overflow
2	Underflow
1	Divide by Zero
0	Inexact

Floating-point queue

This dual-ported register allows for FALU/FMPY instructions to be sent simultaneously with the unloading of the previous operation. The instruction port is referred to as the fire port. The unload port always refers to the top of the queue and, if an exception occurs, the port de-queues unfinished instructions to the external system bus.

FLOATING POINT DATA

The ANSI/IEEE 754-1985 floating point data types supported by the FPC include the definition of single, double, and extended precision. Single precision operands are 32-bits wide, double precision operands are 64-bits wide, and extended precision operands are 128-bits wide. Of the 128-bits reserved for an extended precision word, only 80-bits are used. The remaining 48 unused bits are always associated with an extended operand, however, since all operands must be evenly aligned in memory. BIT SPARC supports Single and Double precision operands in hardware, but extended precision Fpops will generate an Unimplemented_Fpop trap. Extended precision operands must therefore be accomplished in software.

The SPARC architecture defines word order in memory to be most significant to least significant. For example, a double precision word in memory will have its most-significant word (bits-32 to 63) placed in location *n* and its least-significant word (bits-0 to 31) placed in *n+4*. Load double instructions will thus load the upper half of a double precision operand from location *n* into f-register *r* specified in the *rd* field of the instruction, and the lower half will be loaded from *n+4* into f-register *r+1*. The address of a single, double, or extended precision word is the address of its most significant byte.

In addition, all words must be evenly aligned in memory. The address of a single precision word must be divisible by 4, that of a double precision word by 8, and the address of an extended precision word must be divisible by 16. If a load or store instruction generates an improperly aligned address, a *memory address not aligned* trap occurs.

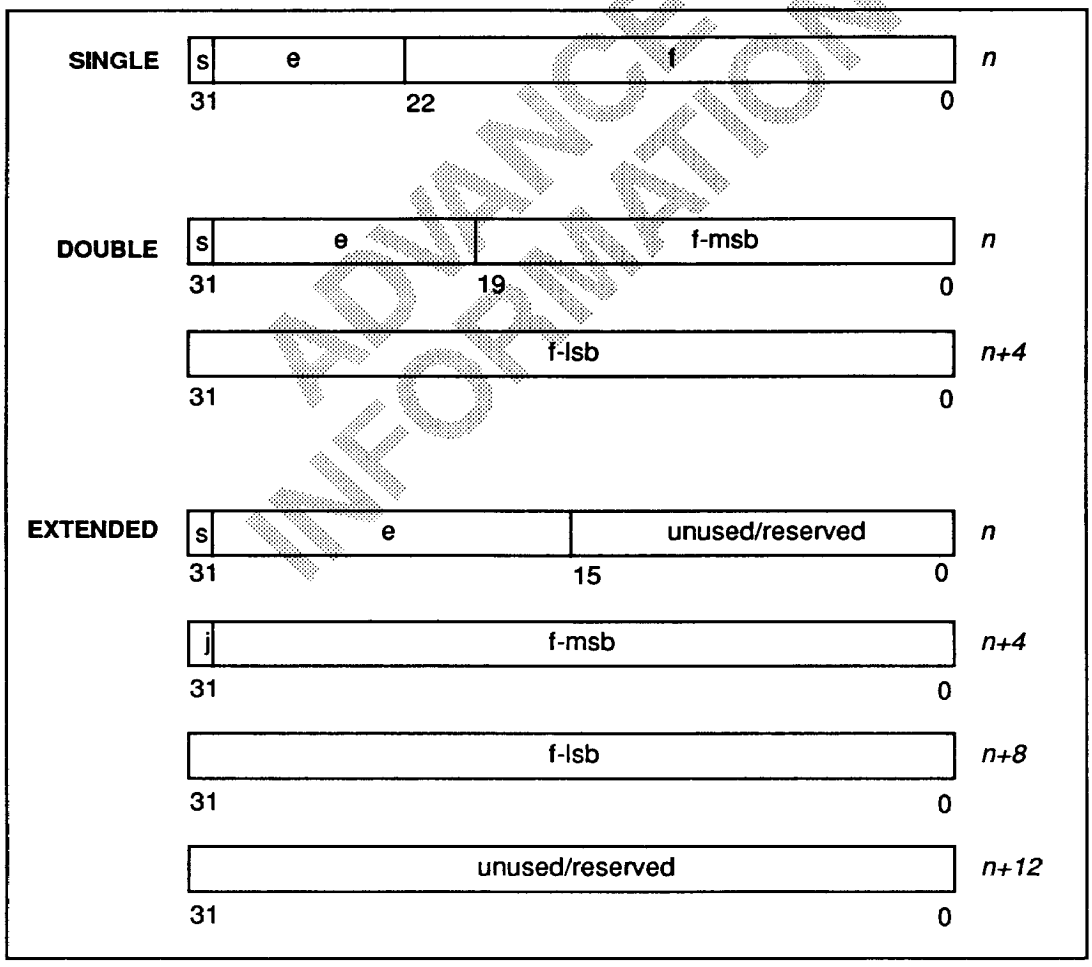


***Figure 3** — Floating Point Data Types*

INSTRUCTION SET

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)*. While these instruction codes are based upon three different formats, only the third format is used for floating point control. It has a 2-bit op field, a 5-bit destination f-register field *rd*, a 6-bit op3 field, a 5-bit source-1 register field *rs1* , a 9-bit FPop field *opf* and a 5-bit source-2 field *rs2*. More information concerning these formats may be found in the SPARC architecture manual. Figure 5 illustrates the Fpop format.

Floating Point 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.

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 will not contain the address of the floating point instruction that caused the exception. As indicated in the section on Traps, however, the first element of the floating point queue points to this instruction, and the rest of the queue contains instructions which have not yet completed.

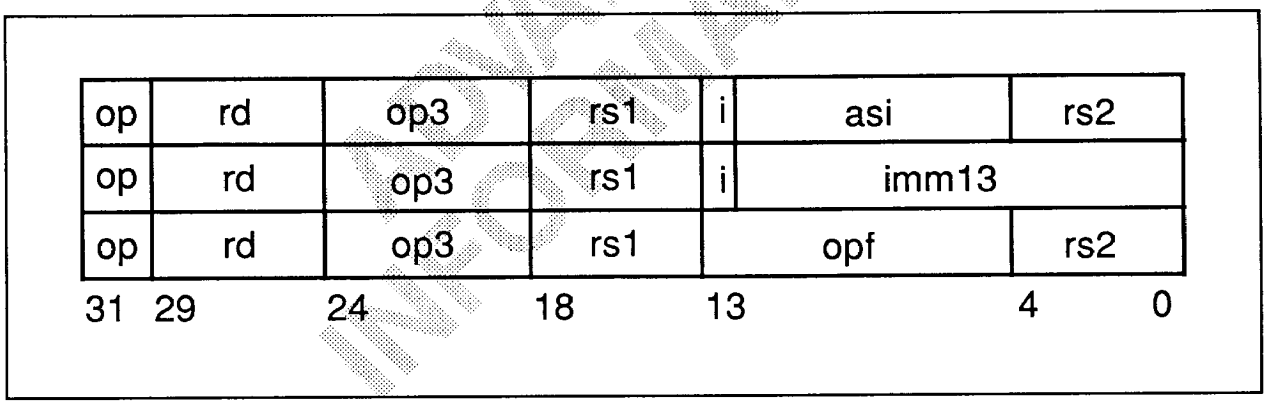


Figure 4 *— FPU Instruction Format*

Table 1 — Instruction Set

Mnemonic	Function	Cycles
FPop1	*Floating-point operate, all instructions except Compare*	
FPop2	*Floating-point operate, Compare instructions*	
FiTOs	*Convert Integer to Single*	
FiTOd	*Convert Integer to Double*	
FiTOx	*Convert Integer to Extended*	
FsTOi	*Convert Single to Integer*	
FdTOi	*Convert Double to Integer*	
FxTOi	*Convert Extended to Integer*	
FsTOiR	*Convert Single to Integer and Round*	
FdTOiR	*Convert Double to Integer and Round*	
FxTOiR	*Convert Extended to Integer and Round*	
FsTOd	*Convert Single to Double*	
FsTOx	*Convert Single to Extended*	
FdTOs	*Convert Double to Single*	
FdTOx	*Convert Double to Extended*	
FxTOs	*Convert Extended to Single*	
FxTOd	*Convert Extended to Double*	
FMOVs	*Move*	
FNEGs	*Negate*	
FABSs	*Absolute Value*	
FSQRTs	*Square Root Single*	
FSQRTd	*Square Root Double*	
FSQRTx	*Square Root Extended*	
FADDs	*Add Single*	
FADDd	*Add Double*	
FADDx	*Add Extended*	
FSUBs	*Subtract Single*	
FSUBd	*Subtract Double*	
FSUBx	*Subtract Extended*	
FMULs	*Multiply Single*	
FMULd	*Multiply Double*	
FMULx	*Multiply Extended*	
FDIVs	*Divide Single*	
FDIVd	*Divide Double*	
FDIVx	*Divide Extended*	
FCMPs	*Compare Single*	
FCMPd	*Compare Double*	
FCMPx	*Compare Extended*	
FCMPEs	*Compare Single and Exception if Unordered*	
FCMPEd	*Compare Double and Exception if Unordered*	
FCMPEx	*Compare Extended and Exception if Unordered*	

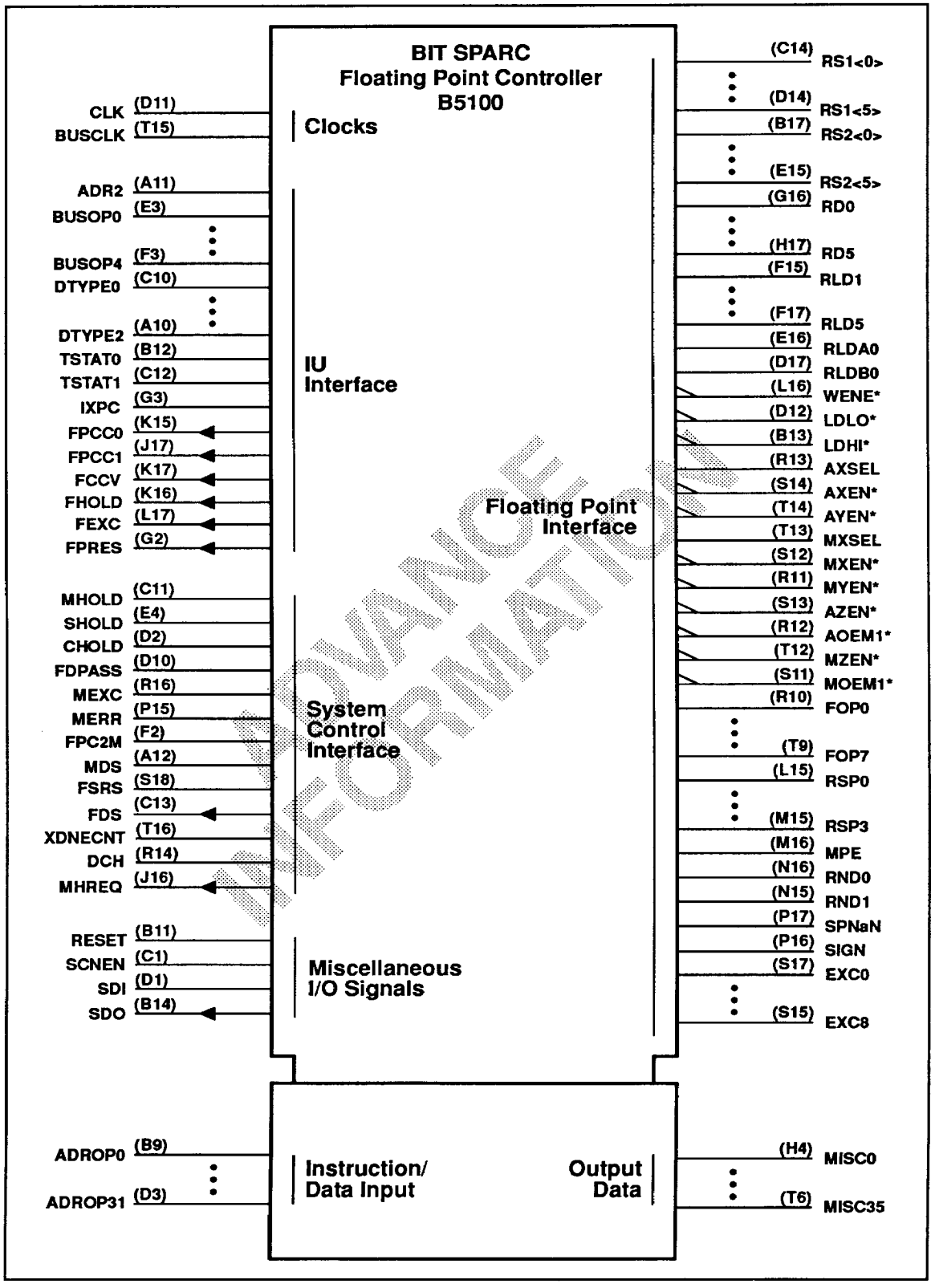


Figure 5 — Pin Assignments

Table 2 — FPU Signal Summary

Clocks		
Main clock	CLK	Input
FAU DP bus clock	BUSCLK	Input
IU Interface		
Instruction/address bus	ADROP<31..0>	Input
Address bit 2	ADR2	Input
Bus operation type	BUSOP<4..0>	Input
Data type indicator	DTYPE<2..0>	Input
Trap status	TSTAT<1..0>	Input
FPC present	FPRES	Output - 50 ohms
Floating point condition codes	FPCC<1..0>	Output - 50 ohms
FCC bits valid	FCCV	Output - 50 ohms
Floating point coprocessor hold	FHOLD	Output - 50 ohms
Floating point exception	FEXC	Output - 50 ohms
Increment external PC	IXPC	
System Control Interface		
Output data	MISC<35..0>	Output - 25 ohms
Hold indication	MHOLD, SHOLD, CHOLD	Input
Data pass through control	FDPASS	Input
Memory exception/error indication	MEXC, MERR	Input
Output bus enable	FPC2M	Input
Data strobe controls	MDS, FSRS	Input
Data strobe output	FDS	Output - 50 ohms
Extended done count	XDNECNT	Input
Disable chaining	DCH	Input
MHOLD request	MHREQ	Output - 50 ohms
Floating Point Interface		
Operand register addresses	RS1<5..0>, RS2<5..0>	Output - 50 ohms
Destination register addresses	RD<5..0>, RLD<5..1>	Output - 50 ohms
Load address LSBs	RLDB0, RLDA0	Output - 50 ohms
Result register write enable	WENE*	Output - 50 ohms
Load data register strobes	LDLO*, LDHI*	Output - 50 ohms
ALU unit load controls	AXSEL, AXEN*, AYEN*	Output - 50 ohms
MPY unit load controls	MXSEL, MXEN*, MYEN*	Output - 50 ohms
ALU unit unload controls	AZEN*, AOEM1*	Output - 50 ohms
MPY unit unload controls	MZEN*, MOEM1*	Output - 50 ohms
Execution unit OP code	FOP<7..0>	Output - 50 ohms
Result bus parity	RSP<3..0>	Output - 50 ohms
State register mode bits	MPE, RD1, RD0, SPNaN, SIGN	Output - 50 ohms
Execution unit exception	EXC<8..0>	Input
Miscellaneous I/O signals		
Reset	RESET	Input
Scan mode enable	SCNEN	Input
Scan data in	SDI	Input
Scan data out	SDO	Output - 50 ohms

Signal Description

Clocks

CLK — Main FPC clock; internal state machines and I/O are based on this clock, typically derived from the IU CLK_OUT. The internal scan path is controlled with CLK.

BUSCLK — Double precision operand load and unload timing for the floating point execution units is controlled with this free running clock. Expected to be one-half the frequency of CLK, this clock is ignored during single precision operations. It is used externally to drive the MSWSEL and MSWEN signals on the B5110 and B5120, and to control the LSB of the result address driven by the FPC.

IU Interface

ADROP<31..0> — Floating point instructions and instruction addresses are received by the FPC over this bus. It should be connected to the IU bus DOUT<31..0>. IU decode stage program counter values are also received from this bus and used to update the on-chip PC.

ADR2 — Address bit 2 from the IU (ALSB<2>). It is used by the FPC to determine which half of the double word bus contains the operand during single precision load operations. This signal is internally latched by the FPC during the current W cycle.

BUSOP<4..0> — This bus is used by the IU to indicate the current operation using the bus. A complete description of the various codes can be found in the IU data sheet. The FPC responds to the following BUSOP codes:

BUSOP<4..0>	Function
00000	NULL
10000	LDF
10001	LDFSR
10011	LDDF
10100	STF
10101	STFSR
10110	STDFQ
10111	STDF

DTYPE<2..0> — Since the signals DTYPE<2..0> indicate the type of data on the IU DOUT bus every cycle, the FPC monitors this bus and responds to the following codes:

DTYPE	Type of data
0	Floating point instruction
3	Decode stage PC
4	Previously transmitted FP instruction
7	NULL

TSTAT<1..0> — System trap status indicators. These pins indicate a floating point trap taken during the R cycle. TSTAT = 01 indicates a floating point trap has been taken by the IU. See the IU data sheet for further information.

FPRES — This signal is generated by the FPC to inform the IU that a floating point coprocessor subsystem is present. FPRES is a static signal, always "1" when an FPC is connected.

FPCC<1..0> — Floating point condition codes are generated on this bus by the FPC. They are set by floating point compare instructions and FSR write operations. They are coded as follows:

FPCC	Relation of fs1,fs2
0	fs1=fs2
1	fs1<fs2
2	fs1>fs2
3	fs1?fs2 (unordered)

FCCV — Floating point condition codes valid indicator to the IU. Condition codes will not be interpreted as valid by the IU until this signal is asserted.

FHOLD — Floating point HOLD signal to the IU. Whenever a conflict exists which causes the IU to be stalled during floating point operations, this signal is used to perform the stall.

FEXC — Floating point exception signal. This signal indicates an IEEE exception, sequence error, or other FPU exception; it will stay active until acknowledged by TSTAT==FPTRAP.

IXPC — Increment external program counter; this control is used by the IU to signal the FPC that its internal copy of the program counter should be incremented. The FPC uses this counter to follow the execution of instructions by the IU. When the IU DOUT bus does not contain the decode PC and IXPC is low, the FPC holds its internal counter.

System Control Interface

MISC<35..0> Peripheral access data bus, used for returning data during STFSR and STDFQ. This bus is enabled by the signal FPC2M.

MHOLD, SHOLD, CHOLD Memory, System and Coprocessor holds issued by the Memory cache controller, auxilliary sources and a coprocessor, respectively.

FDPASS This signal switches the register file to a pass through mode wherein data on the load bus is directly written to the operand bus.

MEXC, MERR Memory exception, memory error. Monitored by the FPC during load or store cycles, these signals indicate a memory exception or error has occurred thus causing the IU/FPC to abort the current memory access.

FPC2M Generated by the cache controller to drive FPC data onto the miscellaneous bus during FSR and STDFQ cycles.

MDS Cache controller data strobe. Reloads the floating point register file during non-cached data or a cache miss fpload. Used during an MHOLD to load.

FSRS FSR data strobe. Issued by the cache controller when FSR data is valid on the ADROP bus during load FSR cycles. This signal always occurs in the cycle following MHOLD negation. If memory exception, the FSR will not be updated and the mode registers of the FALU and FMPY will be unchanged.

FDS This signal from the FPC indicates valid FPC output data is available to the system bus.

XDNECNT Extended done count. Causes the FPC to wait aditional cycles before unloading results from the FALU or the FMPY. The wait, in cycles, depends upon the instruction, as per the following table:

fpop	Unit	XDNECNT ==0		XDNECNT ==1	
		SP	DP	SP	DP
MUL	fmpy	3	4	4	5
DIV	fmpy	14	24	14	24
SQRT	fmpy	24	45	24	45
all	falu	2	2	3	3

DCH Disable chain. Inhibits operand chaining between FALU and FMPY result T ports. This is a static DC signal, normally set to a 10KH low level.

MHREQ MHOLD request. During the W cycle of a load this signal causes an R cycle stall. It is driven by the FPC to the cache controller and the controller responds with an MHOLD until MHREQ goes inactive.

FAU Interface

RS1<5..0> The Op1 source address. The SRC1 address for operand $fregr_{s1}$ to the f-register. Store addresses are multiplexed here for the low word or for the effective address +4.

RS2<5..0> The Op2 source address. The SRC2 address for operand $fregr_{s2}$ to the f-register. Store addresses are multiplexed here for the high word or for the effective address.

RD<5..0> FPreg result address. The address for $freg_{rd}$ which contains an arithmetic result. If a memory exception occurs during load, RD will contain the address to be used for zeroing the load destination register.

RLD<5..1> FPreg load address. During LDF and LDDF instructions this bus drives the load address. RLD lines are common for the *rfile* A and B port addresses.

RLDB0/RLDA0 Bit-0 for PortB/A load address. It is the LSB for PortB and A $freg_{rd}$ addresses, respectively. These bits are controlled by the mapping of the 64-bit load bus to the two floating point registers during LDF and LDDF instructions, as follows:

Load data HI	Load data LO	Destination FP_register	rfile port	RLDB/A0
X		evenBrd0	B	rd0
X		oddBrd0	B	rd0
	X	evenArd0	A	rd0
	X	oddArd0	A	rd0
X	X	even	B & A	0,1

WENE* Write enable, result port. When arithmetic results are driven onto the result bus this signal enables an *rfile* write for port E. During an *ldf* instruction resulting in a memory exception this signal allows clearing of $freg_{rd}$. RD specifies the destination register.

LDHI*/LDLO* LDDF/LDF Hi/Lo data strobe. Write enable for *rfile* ports A and B, respectively. Both are asserted during LDDF and address bit-2 controls these strobes during LDF instructions.

AXSEL/MXSEL FALU/FMPY XA register input mux select. Selects mode register data and chained operands from the T port into the X operand register.

AXEN*/MXEN* FALU/FMPY XA register clock enable. Used for loading data from OP(rs2) and FSR to the mode register.

AYEN*/MYEN* FALU/FMPY YA register clock enable. Used for loading data from OP(rs1) to the mode register.

AZEN*/MZEN* FALU/FMPY Z result register enable.This signal's timing is opcode dependent and can be varied with XDNECNT.

AOEM1*/MOEM1* FALU/FMPY T/F port output enable. Drives results and exceptions to the result bus via the T and F ports, respectively. The FALU/FMPY re-synchronize these signals.

FOP<7..0> FALU/FMPY instruction. The instruction field for performing floating-point operations. Opcode mapping is per Table 3.

RSP<3..0> Result bus parity. These are the odd parity bits when the FPC is driving the result bus (B5110/B5120 T port) for mode register writes on LFSR cycles, and *fpreg* for loads if a memory exception has occurred. These 3 lines are connected to MISC <35..32>.

MPE Mode Parity Enable. On *load fsr* cycles, this is the mode register bit-2. This bit is always asserted.

RD1 Rounding direction bit-1. Driven from *fsr31* on *load fsr* cycles, this is the B5110/B5120 mode register bit-6.

RD0 Rounding direction bit-0. Driven from *(fsr31 xor fsr30)* on *load fsr* cycles, this is the B5110/B5120 mode register bit-5.

SPNaN SPARC NaN encoding. The FALU/FMPY will generate SPARC compatible NaN encodings when this bit is asserted. The output is written during LDFSR instructions to B5110/B5120 mode register bit-9.

SIGN Sign bit. Used to set the MSB of the B5210 register file location-32. This location is used as an operand for absolute value (fabss) and negate (fnegs) Fpops. See table 3.

EXC<8..0> FALU/FMPY exception. When OEM1* is asserted to the respective data path device, these bits specify the exemption status of the arithmetic operation.

EXC <0>: N: Negative result.
EXC <1>: Z: Zero flag.
EXC <2>: OF: Overflow.
EXC <3>: UF: Underflow.
EXC <4>: NV: Invalid operation.
EXC <5>: NX: Inexact result.
EXC <6>: NaN: Not a number.
EXC <7>: DEN: Denormalized operand(s).
EXC <8>: DZ/CRY: Divide by zero or carry flag for FMPY or FALU respectively.

Miscellaneous I/O signals

RESET System reset. Reset can be applied at any time because it is double-ranked synchronized. It forces the FPC into a known state and negates all outputs.

SCNEN Scan mode enable.

SDI Scan data input.

SDO Scan data output.

Table 3 — FOP<7..0> Opcode Mapping

mnemonic	opf	src	FMPY/FALU function	Exception (EXC) Flags				
				NV	OF	UF	DZ	NX
fitos fitod fitox	01100110 01100111	X X	SICF: sp ← 32bit sint SICDF: sp ← 32bit sint	0 0	0 0	0 0	0 0	x 0
fstoi fdtoi fxtoi	01110010 01110011	X X	FCSIT: sp → 32bit sint DFCSIT: sp → 32bit sint	x x	0 0	0 0	0 0	x x
fstod fstox fdtos fdtox fxtos fxtod	01110110 01110111	X X	SDF: sp → dp DSDF: dp → sp	x x	0 x	0 x	0 *	0 x
fmovs fnegs fabss	11011011 11011111 11010100	X Y X Y X Y	IPASSX: rd ← X IXOR: rd ← –X IANDNY: rd ← [X] Y = 0x80000000 (rfile-32)	x x x	0 0 0	x x x	* * *	0 0 0
fsqrts fsqrtd fsqrtx	00000010 00000011	X X	SQRTZ: rd ← sq X DSQRTZ: rd ← sq X	x x	0 0	0 0	0 0	x x
fadds faddd faddx	00110000 00110001	X Y X Y	ADD: X + Y DADD: X + Y	x x	x x	x x	0 0	x x
fsubs fsubd fsubx	00110100 00110010 00110101 00110011	X Y X Y X Y X Y	SUBX: Y – X SUB: X – Y DSUBX: Y – X SUB: X – Y	x x x x	x x x x	x x x x	0 0 0 0	x x x x
fmuls fmuld fmulx	00001000 00001001	X Y X Y	MULT: X * Y DMULT: X * Y	x x	x x	x x	0 0	x x
fdivs fdivd fdivx	00000000 00000001	X Y X Y	DIV: X ! Y DDIV: X ! Y	x x	x x	x x	x x	x x
fcmps fcmps fcmpx	00110110 00110111	X Y X Y	CMPR: X , Y DCMPR: X , Y	x x	0 0	0 0	* *	0 0
fcmpes fcmpes fcmpex	00110110 00110111	X Y X Y	CMPR: X , Y DCMPR: X , Y	x x	0 0	0 0	* *	0 0

* FALU outputs CRY (carry flag) on the same pin that FMUL outputs the DIVZ flag. These instructions update the carry flag and therefore must be masked to zero before updating the FSR.

Electrical Specifications

Table 4 — Absolute Maximum Ratings

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.

Parameter	Symbol	Value		Units
		With Heatsink	Without Heatsink	
Supply Voltage	V_{ee}	–7.0 to 0	-7.0 to 0	V_{dc}
Input Voltage	V_{in}	V_{ee} to 0	V_{ee} to 0	V_{dc}
Output Source Current				
Continuous (50 Ω output)	I_{oc}	30	30	mA_{dc}
Continuous (25 Ω output)	I_{oc}	60	60	mA_{dc}
Surge	I_{os}	100	100	mA_{dc}
Storage Temperature	T_{st}	125	150	°C
Junction Temperature	T_j	125	225	°C

Table 5 — Recommended Operating and Test Conditions

Note: The following environmental conditions pertain to guaranteed DC and Switching characteristics for chips soldered into circuit boards. These conditions should not be exceeded during normal operation.

Parameter	Symbol	Min	Nom	Max	Units
Supply Voltage (V_{cc} = 0)	V_{ee}	–5.46	–5.20	–4.94	V_{dc}
Operating Junction Temperature	T_{jo} *	20		125	°C
Output Termination to –2.0 V_{dc}	R_t **		50		Ω

* Worst case junction temperature specified for worst case I_{ee}

** MISC <35..0> use 25 Ω output termination.

Table 6 — DC Characteristics

Parameter	Symbol	T_{jo} min		T_{jo} nom		T_{jo} max		Units
		Min	Max	Min	Max	Min	Max	
Input Voltage High	V_{ih}	–1.17		–1.13		–1.07		V_{dc}
Input Voltage Low	V_{il}		–1.48		–1.48		–1.45	V_{dc}
Output Voltage High	V_{oh}	–1.02		–0.98		–0.92		V_{dc}
Output Voltage Low	V_{ol}		–1.63		–1.63		–1.60	V_{dc}
MISC<35..0>output disab'd	V_{olz}		–2.0		–2.0		–2.0	V_{dc}

Parameter	Symbol	Min	Nom	Max	Units
Supply Current	I_{ee}		2.75	3.75	A_{dc}
Input Current High	I_{ih}			0.3	mA_{dc}
Input Capacitance	C_{in}		5		pF

Table 7 — Switching Characteristics

Parameter	Symbol	Min	Max	Units
Propagation Delay				
CLK to MISC<35..0>	t_{ckm}	2.0	8.5	ns
CLK to FPCC<1..0>	t_{ckcc}	1.5	8.5	ns
CLK to FCCV	t_{ckcv}	1.5	8.5	ns
CLK to FDS	t_{ckds}	1.0	5.0	ns
CLK to FHOLD	t_{ckhd}	1.5	8.5	ns
CLK to MHREQ	t_{ckrq}	1.0	7.5	ns
CLK to FEXC	t_{ckex}	1.5	8.5	ns
CLK to RS2<5..0>	t_{cks2}	1.0	7.5	ns
CLK to RS1<5..0>	t_{cks1}	1.0	7.5	ns
CLK to RD<5..0>	t_{ckrd}	1.0	7.5	ns
CLK to RLD<5..1>	t_{ckrl}	1.0	7.5	ns
CLK to RLDA0	t_{cka0}	1.0	7.5	ns
CLK to RLDB0	t_{ckb0}	1.0	7.5	ns
CLK to WENE*	t_{ckwe}	2.0	7.5	ns
CLK to LDLO*	t_{cklo}	1.0	7.5	ns
CLK to LDHI*	t_{ckhi}	1.0	7.5	ns
CLK to AXSEL	t_{ckaxs}	1.0	6.5	ns
CLK to AXEN*	t_{ckaxn}	1.0	7.5	ns
CLK to AYEN*	t_{ckayn}	1.0	7.5	ns
CLK to AZEN*	t_{ckazn}	2.0	8.5	ns
CLK to AOEM1*	t_{ckaoe}	2.0	8.5	ns
CLK to MXSEL	t_{ckmxs}	1.0	6.5	ns
CLK to MXEN*	t_{ckmxn}	1.0	7.5	ns
CLK to MYEN*	t_{ckmyn}	1.0	7.5	ns
CLK to MZEN*	t_{ckmzn}	2.0	8.5	ns
CLK to MOEM1*	t_{ckmoe}	2.0	8.5	ns
CLK to FOP<7..0>	t_{ckop}	1.0	7.5	ns
CLK to RSP<3..0>	t_{cksp}	1.0	7.5	ns
CLK to MPE	t_{ckpe}	1.0	7.5	ns
CLK to RND1	t_{ckd1}	1.0	7.5	ns
CLK to RND0	t_{ckd0}	1.0	7.5	ns
CLK to SPNAN	t_{cksn}	1.0	7.5	ns
CLK to SIGN	t_{cksi}	1.0	7.5	ns
CLK to SDO	t_{ckso}	2.0	8.5	ns
TSTAT<1..0> to LDLO*	t_{tslo}	0	4.5	ns
MHOLD to LDLO*	t_{mhlo}	0	4.5	ns
TSTAT<1..0> to LDHI*	t_{tshi}	0	4.5	ns
MHOLD to LDHI*	t_{mhhi}	0	4.5	ns
Output Timing				
Enable, CLK to MISC<35..0>	t_{oe} (ckm)	2.0	8.5	ns
Disable, CLK to MISC<35..0>	t_{dis} (ckm)	2.0	8.5	ns

Table 7 — Switching Characteristics (cont'd)

Parameter	Symbol	Min	Max	Units
Setup–hold Time				
ADROP<35..0> to CLK	t_{su} (adck)	2.5		ns
	t_{hd} (adck)	1.5		ns
ADR2 to CLK	t_{su} (a2ck)	3.0		ns
	t_{hd} (a2ck)	1.0		ns
TSTAT<1..0> to CLK	t_{su} (tpck)	3.5		ns
	t_{hd} (tpck)	0.5		ns
FDPASS to CLK	t_{su} (fdck)	2.5		ns
	t_{hd} (fdck)	1.5		ns
BUSCLK to CLK	t_{su} (mwck)	3.5		ns
	t_{hd} (mwck)	0.5		ns
CHOLD to CLK	t_{su} (chck)	2.0		ns
	t_{hd} (chck)	2.0		ns
SHOLD to CLK	t_{su} (shck)	2.0		ns
	t_{hd} (shck)	2.0		ns
BUSOP<4..0> to CLK	t_{su} (bock)	2.0		ns
	t_{hd} (bock)	2.0		ns
DTYPE<2..0> to CLK	t_{su} (dtck)	3.5		ns
	t_{hd} (dtck)	0.5		ns
MHOLD to CLK	t_{su} (mhck)	3.5		ns
	t_{hd} (mhck)	0.5		ns
IXPC to CLK	t_{su} (ixck)	3.5		ns
	t_{hd} (ixck)	0.5		ns
XDNECNT to CLK	t_{su} (xdck)	3.5		ns
	t_{hd} (xdck)	0.5		ns
DCH to CLK	t_{su} (chck)	3.5		ns
	t_{hd} (chck)	0.5		ns
FPC2M to CLK	t_{su} (fmck)	2.0		ns
	t_{hd} (fmck)	2.0		ns
MEXC to CLK	t_{su} (mxck)	2.0		ns
	t_{hd} (mxck)	2.0		ns
MERR to CLK	t_{su} (meck)	2.0		ns
	t_{hd} (meck)	2.0		ns
MDS to CLK	t_{su} (dsck)	2.0		ns
	t_{hd} (dsck)	2.0		ns
FSRS to CLK	t_{su} (fsck)	2.0		ns
	t_{hd} (fsck)	2.0		ns
EXC<8..0> to CLK	t_{su} (exck)	2.0		ns
	t_{hd} (exck)	2.0		ns
SCNEN to CLK	t_{su} (snck)	2.0		ns
	t_{hd} (snck)	2.0		ns
SDI to CLK	t_{su} (sick)	2.0		ns
	t_{hd} (sick)	2.0		ns
RESET to CLK	t_{su} (rsck)	2.0		ns
	t_{hd} (rsck)	2.0		ns
Miscellaneous Timing				
Cycle time	t_{cyc}	12.5		ns
Clock Low time	t_{wckl}	5.5		ns
Clock High time	t_{wckh}	5.5		ns
Minimum reset pulse width	t_{wrs}	10		clk period

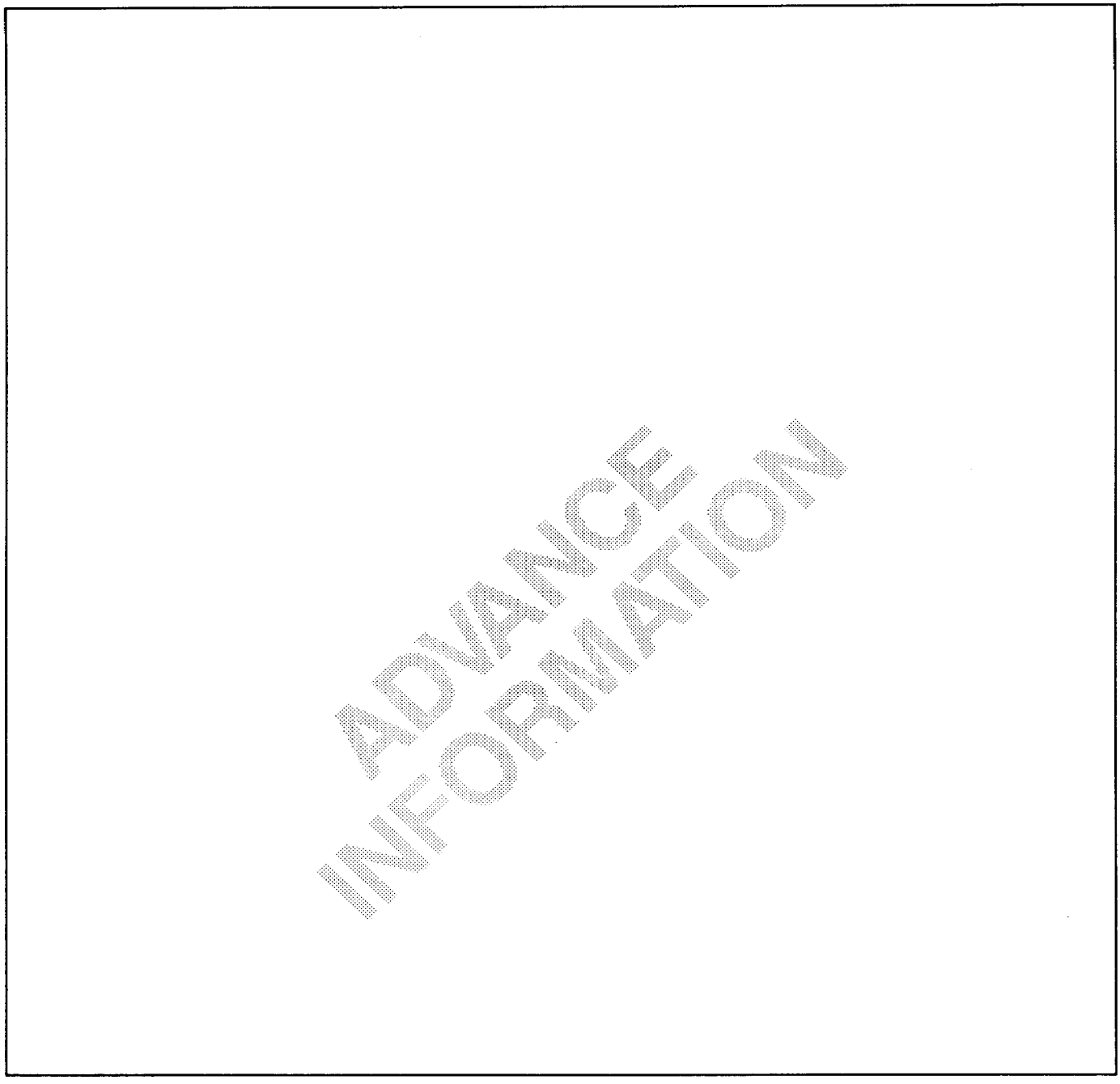


Figure 6 *— Timing Diagrams*

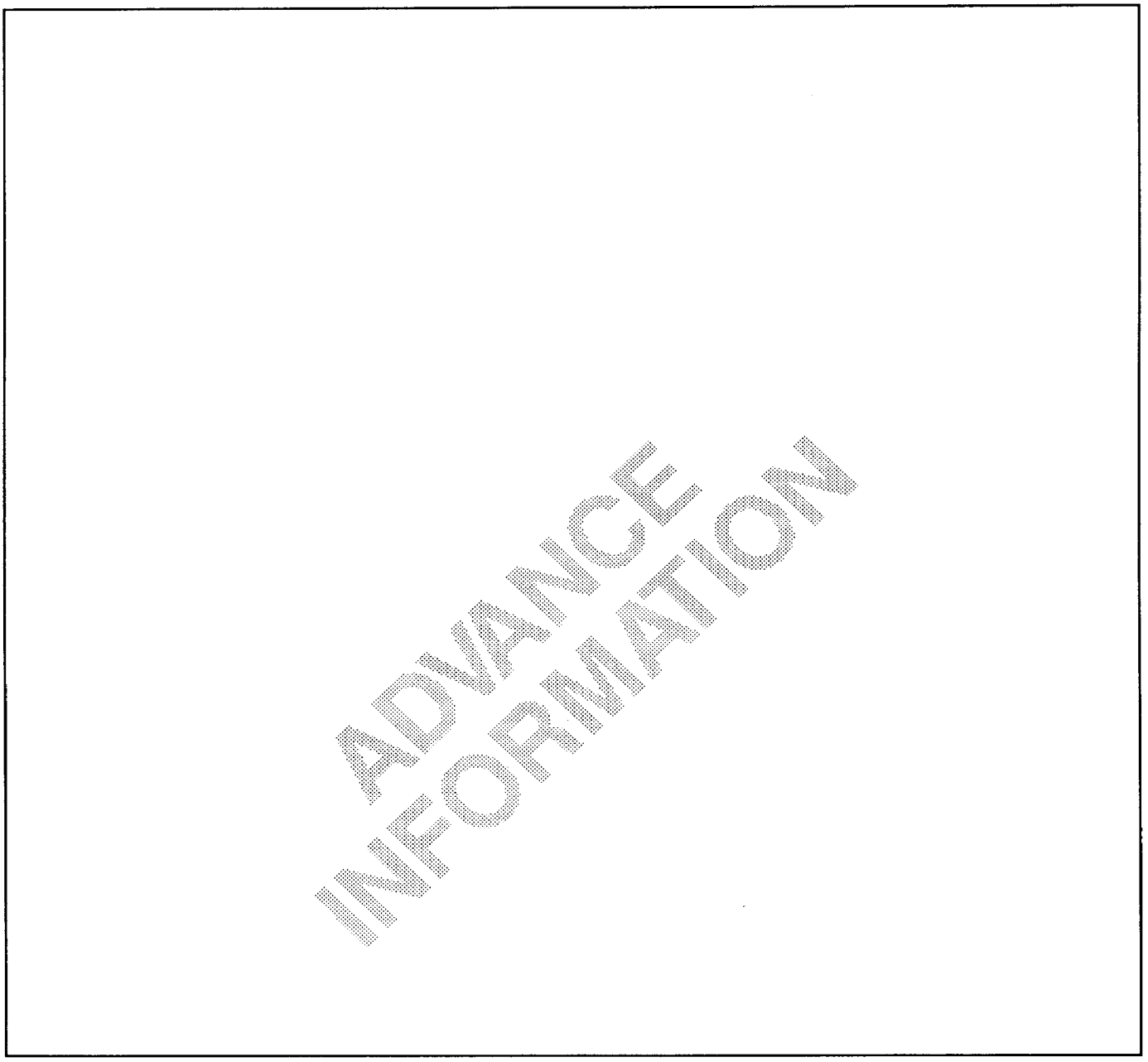


Figure 7 — Timing

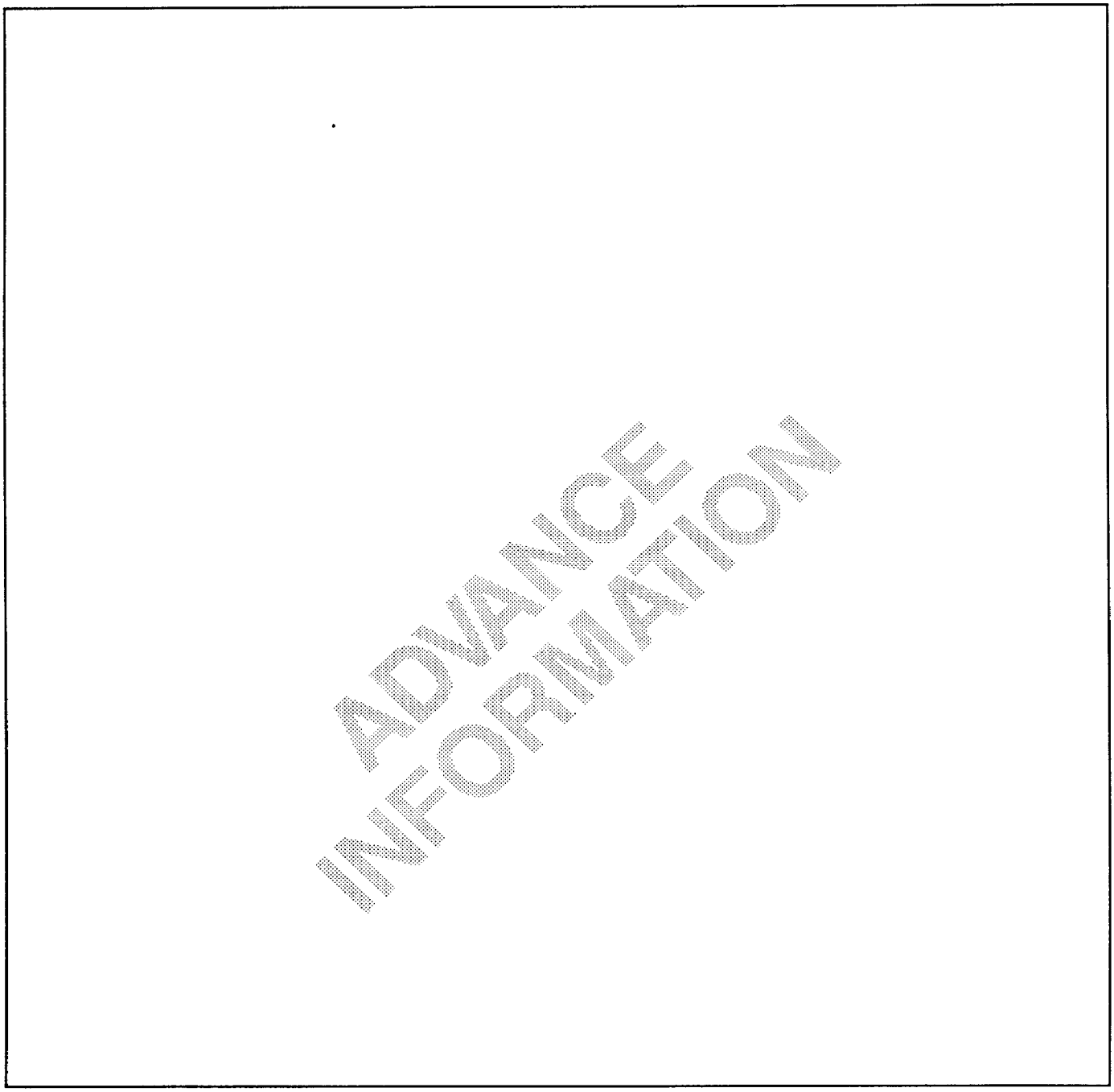


Figure 8— *Timing*

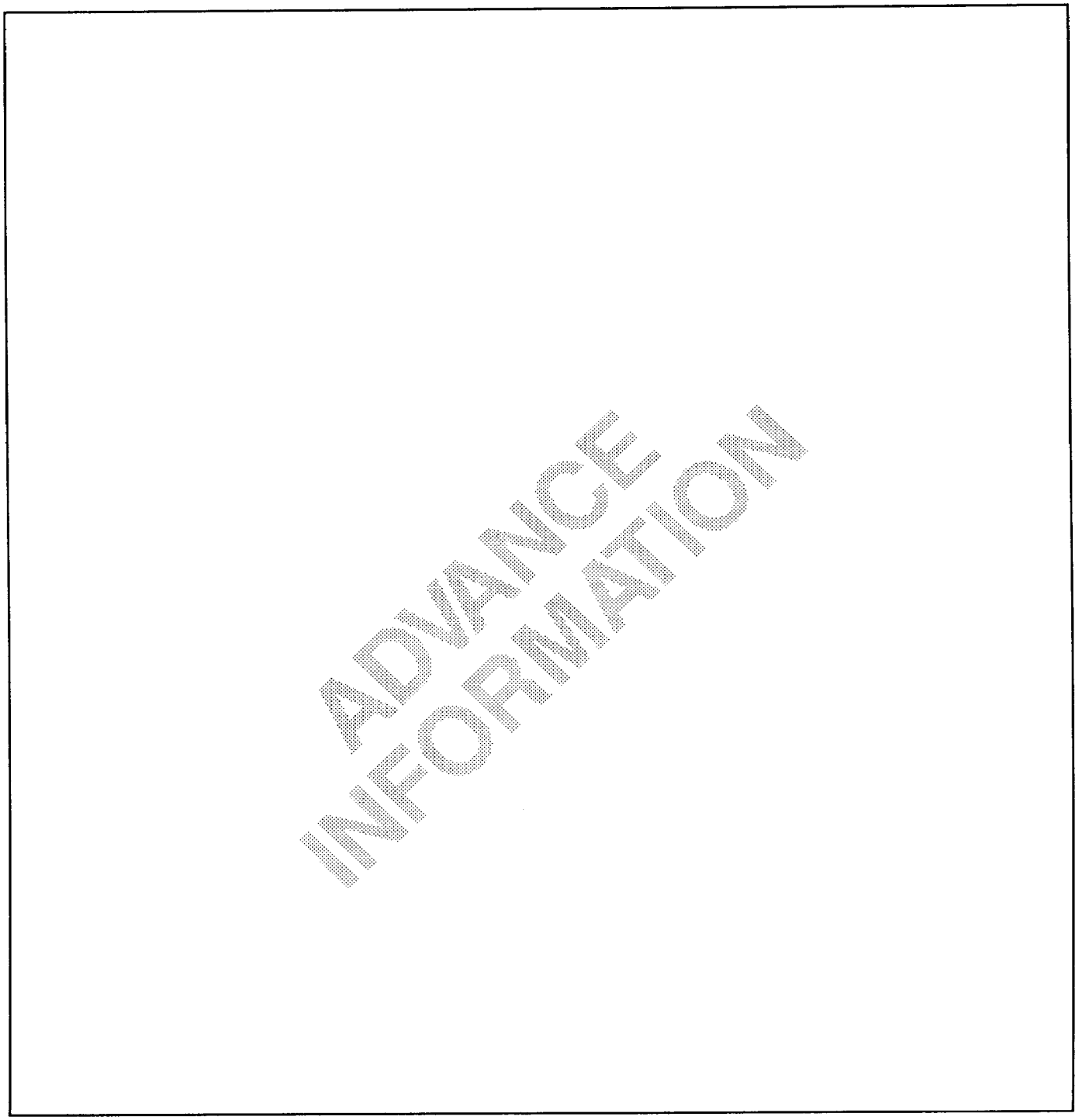


Figure 9 *— Timing*

	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
A	VC0	VC0	VC0	VC0	VC0	VC0	MDS	ADR2	DTYPE (2)	ADROP (1)	ADROP (5)	ADROP (9)	ADROP (12)	ADROP (14)	ADROP (18)	ADROP (22)	ADROP (26)	NO PIN	A
B	VC0	RS2 (0)	RS1 (3)	RS1 (1)	SDO	LDHI*	TSTAT (0)	RESET	DTYPE (1)	ADROP (0)	ADROP (4)	ADROP (8)	ADROP (13)	ADROP (17)	ADROP (21)	ADROP (23)	ADROP (25)	ADROP (30)	B
C	VC0	RS2 (3)	RS2 (1)	RS1 (4)	RS1 (0)	FDS	TSTAT (1)	MHOLD	DTYPE (0)	ADROP (3)	ADROP (7)	ADROP (11)	ADROP (16)	ADROP (20)	ADROP (24)	ADROP (28)	ADROP (29)	SCNEN	C
D	VC0	RLDB0	RS2 (4)	RS2 (2)	RS1 (5)	RS1 (2)	LDLO*	CLK	FDP RES	ADROP (2)	ADROP (6)	ADROP (10)	ADROP (15)	ADROP (19)	ADROP (27)	ADROP (31)	CHOLD	SDI	D
E	VC0	RLD (2)	RLDA0	RS2 (5)	VCC	VCC	VEE	VEE	VCC	VCC	VEE	VEE	VCC	VCC	SHOLD	BUSOP (0)	BUSOP (1)	BUSOP (2)	E
F	VC0	RLD (5)	RLD (3)	RLD (1)	VCC									VCC	BUSOP (3)	BUSOP (4)	FPC2M	N/C	F
G	VC0	RD (2)	RD (0)	RLD (4)	VEE									VEE	N/C	IXPC	FPRES	VC0	G
H	VC0	RD (5)	RD (3)	RD (1)	VEE									VEE	MISC (0)	MISC (1)	MISC (3)	VC0	H
J	VC0	FPCC (1)	MH REQ	RD (4)	VCC									VCC	MISC (4)	MISC (6)	MISC (2)	VC0	J
K	VC0	FCCV	FHOLD	FPCC (0)	VCC									VCC	MISC (9)	MISC (8)	MISC (5)	VC0	K
L	VC0	FEXC	WENE*	RSP (0)	VEE									VEE	MISC (12)	MISC (10)	MISC (7)	VC0	L
M	VC0	RSP (1)	MPE	RSP (3)	VEE									VEE	MISC (15)	MISC (13)	MISC (11)	VC0	M
N	VC0	RSP (2)	RND (0)	RND (1)	VCC									VCC	MISC (19)	MISC (17)	MISC (14)	VC0	N
P	VC0	SPNAN	SIGN	MERR	VCC	VCC	VEE	VEE	VCC	VCC	VEE	VEE	VCC	VCC	MISC (18)	MISC (20)	MISC (16)	VC0	P
R	VC0	VC0	MEXC	EXC (5)	DCH	AXSEL	AOEM1*	MYEN*	FOP (0)	FOP (5)	MISC (32)	MISC (29)	MISC (27)	MISC (26)	MISC (24)	MISC (21)	VC0	VC0	R
S	FSRS	EXC (0)	EXC (6)	EXC (8)	AXEN*	AZEN*	MXEN*	MOEM1*	FOP (4)	FOP (2)	MISC (34)	MISC (30)	MISC (33)	MISC (28)	MISC (25)	MISC (22)	VC0	VC0	S
T	EXC (1)	EXC (2)	XDNE CNT	BUS CLK	AYEN*	MXSEL	MZEN*	FOP (1)	FOP (6)	FOP (7)	FOP (3)	N/C	MISC (35)	MISC (31)	VC0	VC0	MISC (23)	VC0	T
V	EXC (3)	EXC (4)	EXC (7)	VC0	VC0	VC0	VC0	VC0	VC0	VC0	VC0	VC0	VC0	VC0	VC0	VC0	VC0	VC0	V
	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	

Bottom View

Figure 10 — B5100 Pin Out

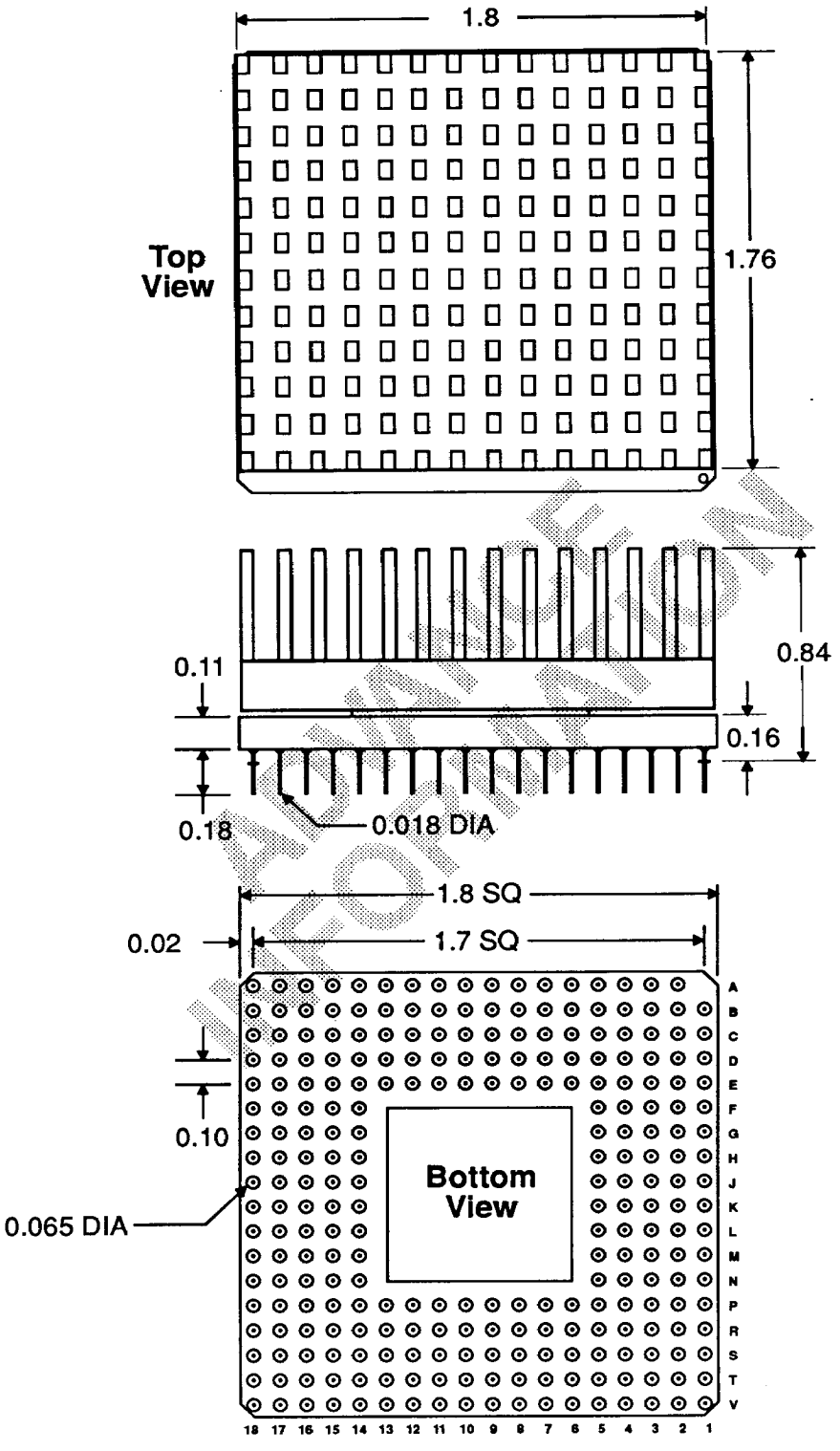


Figure 11 — B5100 Package Dimensions

Sales Offices

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

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

Northeast Sales Office:
50 Mall Road, Suite #201
Burlington, MA 01803
(617) 229-0150

Southeast Sales Office:
214 Parkway 575
Woodstock, GA 30180
(404) 591-2285

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

MKTG-D011
24