WEITEK

ABACUS 3171 FLOATING POINT COPROCESSOR FOR SPARC

PRELIMINARY DATA

May 1989

The Abacus 3171 is a single-chip floating-point coprocessor for the Cypress 7C601 implementation of the SPARC architecture. It incorporates a floating-point datapath and a floating-point controller. The Abacus 3171 provides direct interface to the integer unit and memory. It is available in speed grades of 25, 33 and 40 MHz.

Related product: The Abacus 3170 single-chip floating-point coprocessor for the Fujitsu S-20/S-25 and LSI Logic L64801 implementations of the SPARC architecture.

Contents

USING THIS DATA SHEET

In the writing of this data sheet, it was assumed that the user is familiar with the SPARC architecture as well as the hardware details of its implementation. This data sheet does not cover details that are explained in the following and other related literature:

The SPARC Architecture Manual, by Sun Microsystems

RISC 7C600 Family Users Guide, by Cypress Semiconductor Corporation

WEITEK Abacus 3171 Floating-Point Coprocessor for SPARC
May, 1989

1060 East Arques Avenue
Sunnyvale, California 94086
Telephone (408) 738-8400

WEITEK reserves the right to make changes to these specifications at any time

Printed in the United States of America
90 89 6 5 4 3 2 1

DOC 8949

ABACUS 3171 FLOATING-POINT COPROCESSOR FOR SPARC

PRELIMINARY DATA
May 1989

Features

SINGLE-CHIP 64-BIT FLOATING-POINT DATA PATH AND CONTROLLER

64-bit multiplier and divide/square root unit
64-bit ALU
16×64 or 32×32 three-port register file with an independent load/store port

DIRECT INTERFACE TO CYPRESS 7C601 SPARC PROCESSOR

DIRECT INTERFACE TO MEMORY

25, 33, AND 40 MHz OPERATION

FULL COMPLIANCE WITH ANSI/IEEE–754 STANDARD FOR BINARY FLOATING-POINT ARITHMETIC

143-PIN PGA PACKAGE

LOW-POWER CMOS

Description

The Abacus 3171 is a high-performance, single-chip floating-point coprocessor for the Cypress 7C601 implementation of the SPARC architecture. It incorporates a floating-point datapath and a floating-point controller. The Abacus 3171 provides direct interface to the integer unit and memory. It is available in speed grades of 25, 33, and 40 MHz.

The floating-point datapath circuitry contains a 64-bit multiplier, a 64-bit ALU, a 64-bit divide/square root unit, and a 16-word by 64-bit (or 32-word by 32-bit) three-port register file.

The floating-point controller circuitry handles IEEE exceptions and the interface between the floating-point datapath and the integer unit, as well as between the datapath and memory.

CONFORMANCE TO SPARC ARCHITECTURE

The Abacus 3171 processes instructions within the specifications of the SPARC architecture.

DATA TYPES

The SPARC architecture specifies four data types that can be used in conjunction with the floating-point unit (FPU):

32-bit two's complement integer
single-precision floating-point
double-precision floating-point
extended-precision floating-point

The Abacus 3171 supports all of these data types except extended-precision. Any operation specifying extended-precision data types will be trapped to system software. See the section *Unimplemented Instructions*.

INSTRUCTIONS

During an instruction fetch, both the IU and FPU receive the instruction from the data bus. The IU always starts decoding the instruction once it is fetched. The FPU however, waits until it is signalled by the IU that the fetched instruction is a correct one and therefore should be executed.

When the IU receives a floating-point instruction (FPop), it signals the FPU using FINS1/FINS2 to start execution. Since the two chips are fully synchronized, at any given time the instruction in the decode stage of the FPU is the same instruction as in the decode stage of the IU. The FPU checks for dependencies with currently executing FPops, and if a dependency exists, it asserts FHOLD– signal.

The FPU maintains a queue, called the floating-point queue; it accepts instructions and sends them to the datapath for execution. Each entry in the queue consists of a floating-point instruction and its address. Instructions and their addresses are captured directly from the system address and data buses by the FPU.

CONFORMANCE TO ANSI/IEEE–754 SPECIFICATION FOR BINARY FLOATING POINT ARITHMETIC

The Abacus 3171 conforms to the requirements of the ANSI/IEEE–754 specification.

FLOATING-POINT REGISTER (FSR)

The *SPARC Architecture Manual* contains detailed information about the Floating-Point State Register (FSR). Bits 19:17 of the FSR comprise the version field. The version field specifies the particular floating-point unit/controller implementation. In the case of the 3171, FSR (19:17) = 011_2.

Description, continued

IMPLEMENTED INSTRUCTIONS

Operations involving NaNs and denormalized numbers
require system software assistance or intervention.
They terminate with trap type unfinished.

Mnemonic(s)		Operation
ldf		Load floating-point register
lddf		Load double floating-point register
ldfsr		Load floating-point status register
stf		Store floating-point register
stdf		Store double floating-point register
stfsr		Store floating-point status register
stdfq		Store double floating-point queue
fitos	fitod	convert integer to floating-point (rounded as per *fsr.rd*) (single/double)
fstoi	fdtoi	convert floating-point to integer (rounded toward zero) (single/double)
fstod	fdtos	convert single to double/double to single floating-point
fmovs		register to register move
fnegs		register to register move with sign bit inverted
fabss		register to register move with sign bit set to 0
fsqrts	fsqrtd	floating-point square root (single/double)
fadds	faddd	floating-point add (single/double)
fsubs	fsubd	floating-point subtract (single/double)
fmuls	fmuld	floating-point multiply (single/double)
fdivs	fdivd	floating-point divide (single/double)
fcmps	fcmpd	floating-point compare (single/double)
fcmpes	fcmped	floating-point compare and exception if unordered (single/double)

Figure 1. Implemented instructions

UNIMPLEMENTED INSTRUCTIONS

Mnemonic(s)		Operation
fitox		convert integer to extended floating-point (rounded as per *fsr.rd*)
fxtoi		convert extended floating-point to integer (rounded toward zero)
fxtos	fxtod	convert extended floating-point to single/double floating-point
fstox	fdtox	convert single/double floating-point to extended floating-point
fsqrtx		floating-point square root (extended-precision)
faddx		floating-point add (extended-precision)
fsubx		floating-point subtract (extended-precision)
fmulx		floating-point multiply (extended-precision)
fdivx		floating-point divide (extended-precision)
fcmpx		floating-point compare (extended-precision)
fcmpex		floating-point compare and exception if unordered (extended-precision)
fsmuld		single product to double
fdmulx		double product to extended

Figure 2. Unimplemented instructions

Description, continued

DEVICE DESCRIPTION

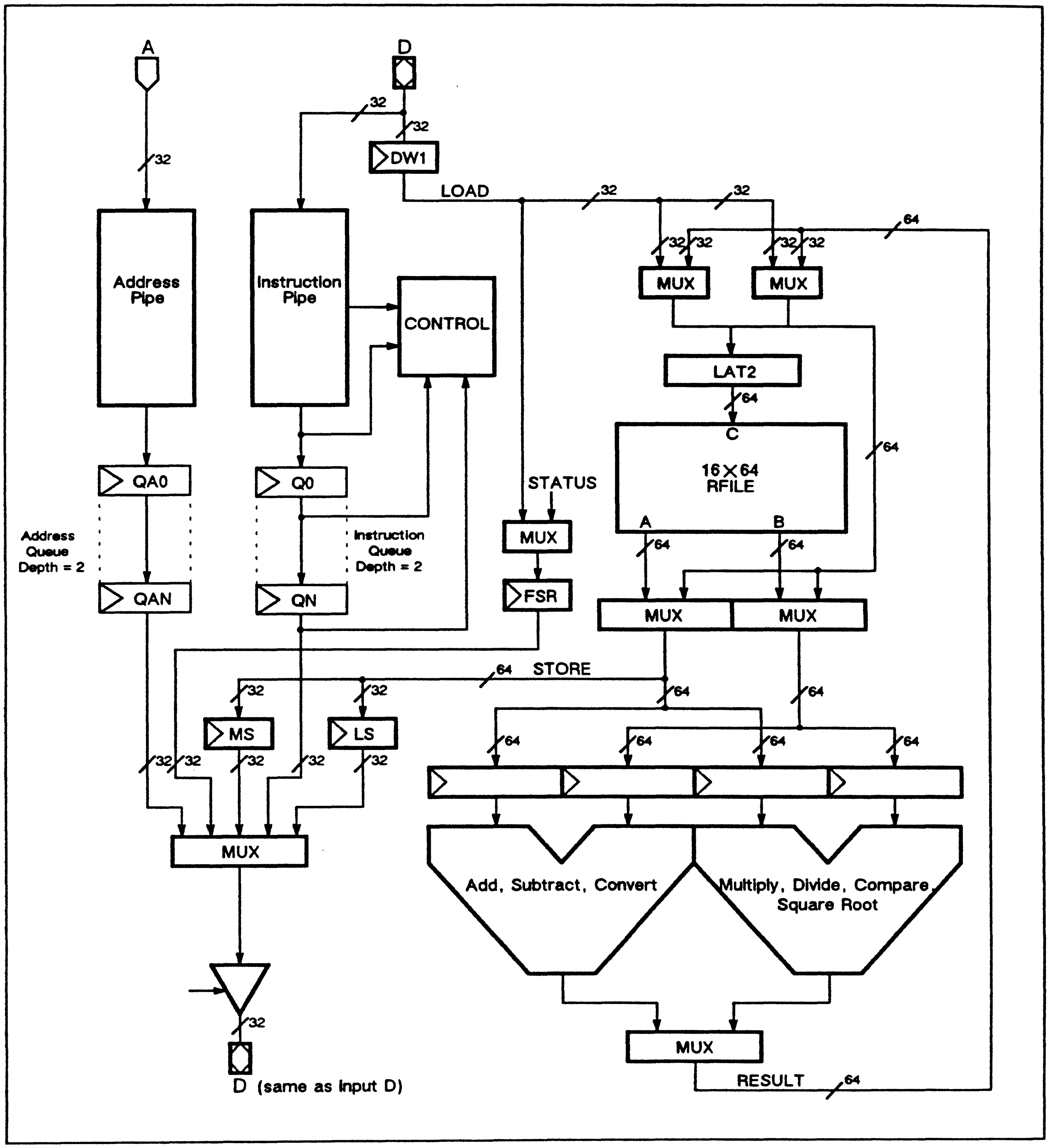


Figure 3. Conceptual block diagram

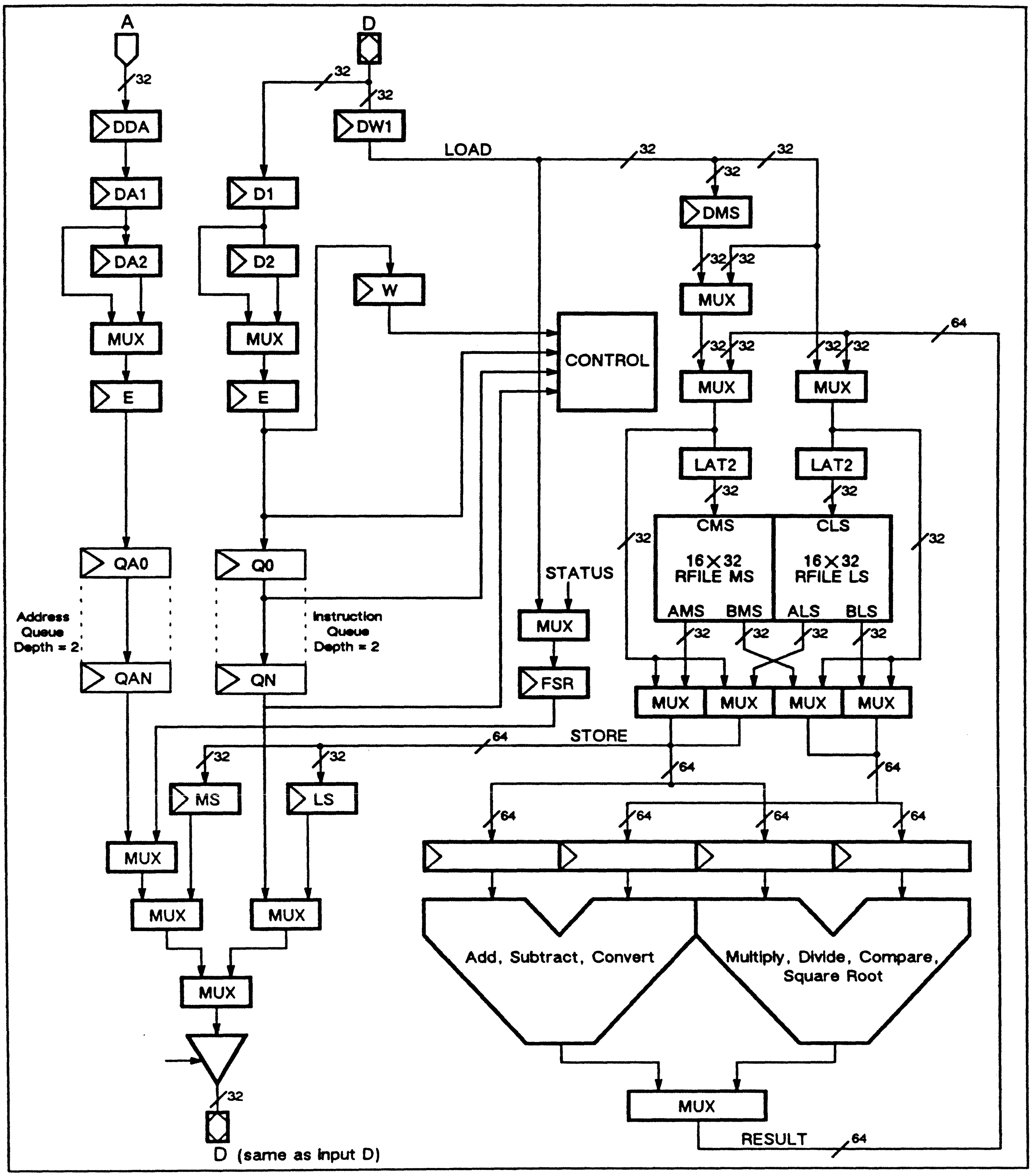


Figure 4. Simplified block diagram

Description, continued

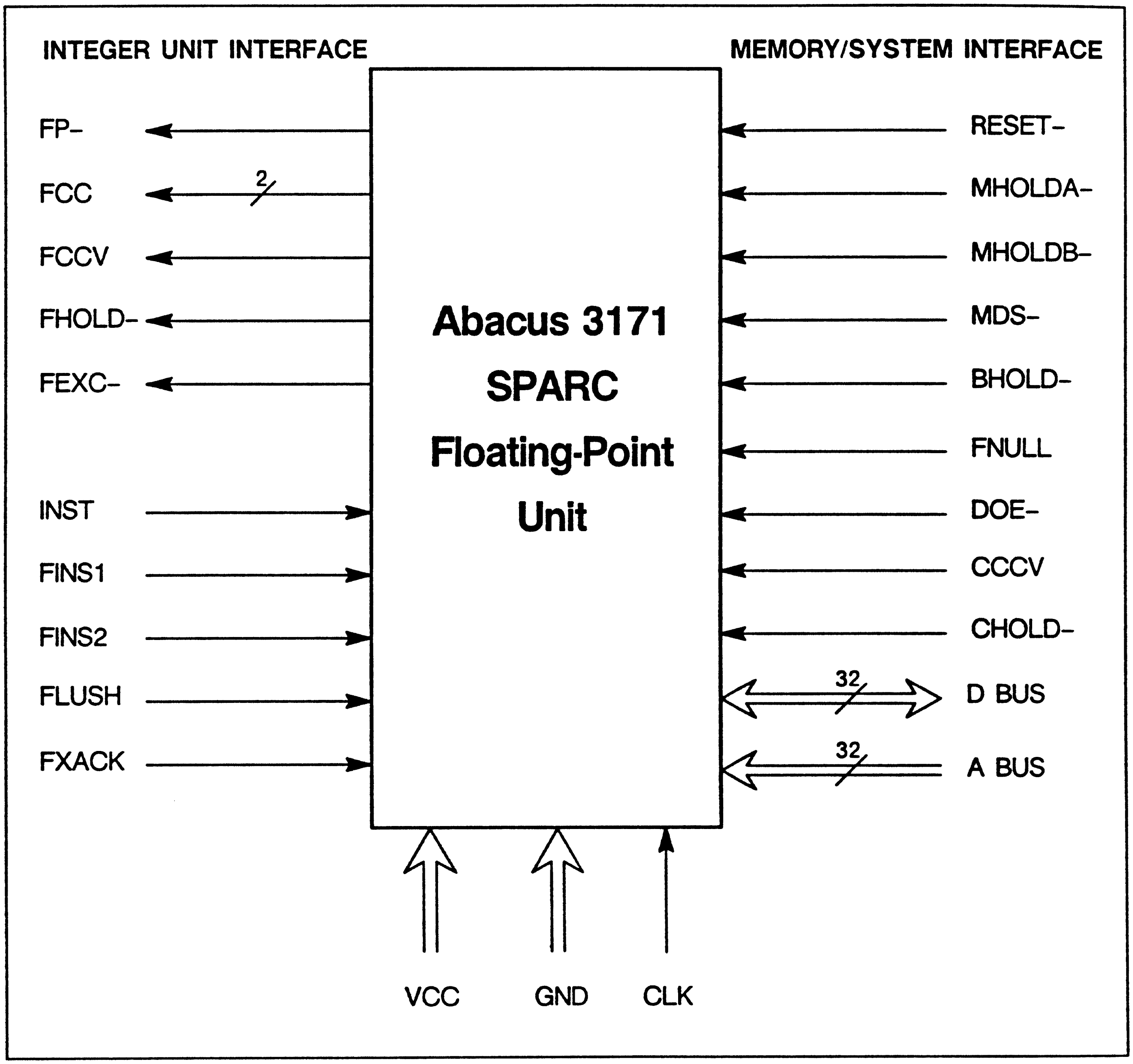


Figure 5. Abacus 3171 signals

Description, continued

SIGNAL DESCRIPTION

Signals marked with a minus sign (–) after their names are active low, all other signals are active high.

INTEGER UNIT INTERFACE SIGNALS

FP– OUTPUT

Floating point unit present. The FP– signal indicates whether a floating-point unit (FPU) is present in the system. In the absence of an FPU the FP– signal is pulled up to VCC by a resistor. When an FPU is present the FP– signal is grounded. The integer unit (IU) generates a floating-point disable trap if FP– is deasserted during the execution of a floating-point instruction, FBfcc instruction, or floating-point load or store instructions.

FCC OUTPUT

Floating-point condition code. The FCC$_{1..0}$ bits represent the current condition code of the FPU. They are valid only if FCCV is asserted.

FBfcc instructions use these bits during the execute cycle if they are valid; and delays the execute cycle if they are not valid. The condition codes are shown below.

FCC (1)	FCC (0)	CONDITION
0	0	Equal
0	1	Op1 < Op2
1	0	Op1 > Op2
1	1	Unordered

Figure 6.

FCCV OUTPUT

Floating-point condition code valid. The FPU asserts the FCCV signal when FCC bits represent a valid condition. The FPU deasserts FCCV if pending floating-point compare instructions exist in the floating-point queue. FCCV is reasserted when the compare instruction is completed and FCC bits are valid.

FHOLD– OUTPUT

Floating-point hold. The FHOLD– signal is asserted by the FPU if it cannot continue execution due to a resource or operand dependency. The FPU checks for all dependencies in the decode stage and, if necessary, asserts FHOLD– in the next cycle. The FHOLD– signal is used by the IU to freeze its pipeline in the same cycle. The FPU must eventually deassert FHOLD– to release the IU's pipeline.

FEXC– OUTPUT

Floating-point exception. The FEXC– signal is asserted if a floating-point exception has occurred. It remains asserted until the IU acknowledges that it has taken a trap by asserting FXACK. Floating-point exceptions are taken only during the execution of a floating-point instruction, FBfcc instruction, or floating-point load or store instructions. When the FPU receives an asserted level of the FXACK signal it deasserts FEXC–.

FXACK INPUT

Floating-point exception acknowledge. The FXACK signal is asserted by the IU to acknowledge to the FPU that the current FEXC– trap is taken.

INST INPUT

Instruction fetch. The INST signal is asserted by the IU whenever a new instruction is being fetched. It is used by the FPU to latch the instruction on the D$_{31..0}$ bus into the FPU instruction buffer. The FPU has two instruction buffers (D1 and D2) to save the last two fetched instructions. When INST is asserted the new instruction enters the D1 buffer and the old instruction in D1 enters the D2 buffer.

FINS1 INPUT

Floating-point instruction in buffer 1. The FINS1 signal is asserted by the IU during the decode stage of an FPU instruction if the instruction is in the D1 buffer of the FPU chip. The FPU uses this signal to latch the instruction in D1 buffer into its execute stage instruction register.

FINS2 INPUT

Floating-point instruction in buffer 2. The FINS2 signal is asserted by the IU during the decode stage of an FPU instruction if the instruction is in the D2 buffer of the FPU chip. The FPU uses this signal to latch the instruction in D2 buffer into its execute stage instruction register.

FLUSH INPUT

Floating-point instruction flush. The FLUSH signal is asserted by the IU to signal to the FPU to flush the instructions in its instruction registers. This may happen when a trap is taken by the IU. The IU will restart the flushed instructions after returning from the trap. FLUSH has no effect on instructions in the floating-point queue. In addition to freezing the pipeline, the FPU uses FLUSH to shut off D bus drivers during store. Therefore, for correct operation of the FPU, when FLUSH is changing

Description, continued

state, it must not switch more than once in any given cycle. This means that FLUSH has to come out of a register triggered on a positive clock edge.

COPROCESSOR INTERFACE SIGNALS

CHOLD– INPUT

Coprocessor hold. The CHOLD– signal is asserted by the coprocessor if it cannot continue execution. The coprocessor checks all dependencies in the decode stage of the instruction and asserts CHOLD– if necessary in the next cycle. This signal is used by the IU and FPU to freeze the instruction in the same cycle. The coprocessor must eventually deassert this signal to unfreeze the FPU's (and the integer unit's) pipeline. The CHOLD– is latched (transparent latch) in the FPU before it is used.

CCCV INPUT

Coprocessor condition codes valid. The coprocessor asserts the CCCV signal when coprocessor condition code CCC(1–0) bits are valid. The coprocessor deasserts CCCV if pending coprocessor compare instructions exist in the coprocessor queue. CCCV is reasserted when the compare instruction is completed and CCC bits are valid. The FPU will enter a wait state if CCCV is deasserted. The CCCV signal is latched (transparent latch) in the FPU before it is used.

SYSTEM/MEMORY INTERFACE SIGNALS

A BUS INPUT/OUTPUT

Address bus. The $A_{31..0}$ is a bus that supplies addresses for instructions and data. The FPU captures addresses of floating-point instructions from the A bus into the DDA register. When INST is asserted, the contents of DDA is transferred to the DA1 register.

D BUS INPUT/OUTPUT

Data bus. The $D_{31..0}$ is a bus that is driven by the FPU only during the execution of floating-point store instructions. The store data is sent out unlatched and must be latched externally before it is used. Once latched, store data is valid during the second data cycle of a store single access, the second and third data cycle of a store double access. The alignment for load and store instructions is done inside the FPU. A double word is aligned on an 8-byte boundary, a word is aligned on a 4-byte boundary.

DOE– INPUT

Data output enable. The DOE– is a signal connected directly to the data output drivers and must be asserted during normal operation. Deassertion of this signal tri-states all output drivers on the data bus. This signal should be deasserted only when the bus is granted to another bus master, i.e., when either BHOLD–, MHOLDA– or MHOLDB– is asserted.

MHOLDA–, MHOLDB– INPUTS

Memory hold. Asserting MHOLDA– or MHOLDB– freezes the FPU pipeline. Either MHOLDA– or MHOLDB– is used to freeze the FPU (and the IU) pipelines during a cache miss (for systems with cache) or when slow memory is accessed.

BHOLD– INPUT

Bus hold. The BHOLD– signal is asserted by the system's I/O controller when an external bus master requests the data bus. Assertion of this signal will freeze the FPU pipeline. External logic should guarantee that after deassertion of BHOLD–, the data at all inputs to the chip is the same as what it was before BHOLD– was asserted.

MDS– INPUT

Memory data strobe. The MDS– signal is used to load data into the FPU when the internal FPU clock is stopped while on hold.

Description, continued

FNULL OUTPUT

FPU nullify cycle. This pin signals to the memory system when the FPU is holding the instruction pipeline of the system. This hold would occur when FHOLD- or FCCV is asserted. This signal is used by the memory system in the same fashion as the integer unit's INULL signal. The system needs this signal because the IU's INULL does not take into account holds requested by the FPU.

RESET- INPUT

Reset. Asserting the RESET- signal resets the pipeline and sets the writable fields of the floating-point status register (FSR) to zero. The RESET- signal must remain asserted for a minimum of eight cycles. After a reset, the IU will start fetching from address 0.

CLK INPUT

Clock. The CLK signal is used for clocking the FPU's pipeline registers. It is high during the first half of the processor cycle and low during the second half. The rising edge of CLK defines the beginning of each pipeline stage in the FPU.

VCC

Power supply. All VCC pins must be connected to 5.0 volt power supply.

GND

System ground. All GND pins must be connected to system ground.

NC

No connection. All NC pins must remain unconnected.

Description, continued

SYSTEM CONSIDERATIONS

LINPACK BENCHMARK ESTIMATE

The code shown below represents the inner loop of the SAXPY subroutine of the LINPACK benchmark. This loop requires 52 cycles on the Abacus 3171. At 33 MHz, this translates into a peak performance of 5.13 MFLOPS; at 40 MHz, 6.15 MFLOPS.

```
loop_top:
    ldd         [dx+0],dx0
    fmuld       dx0,da,dx0
    ldd         [dy+0],dy0
    ldd         [dx+8],dx1
    faddd       dx0,dy0,dy0
    fmuld       dx1,da,dx1
    ldd         [dy+8],dy1
    ldd         [dy+16],dx2
    faddd       dx1,dy1,dy1
    fmuld       dx2,da,dx2
    ldd         [dy+16],dy2
    ldd         [dx+24],dx3
    faddd       dx2,dy2,dy2
    fmuld       dx3,da,dx3
    ldd         [dy+24],dy3
    std         dy0,[dy+0]
    std         dy1,[dy+8]
    faddd       dx3,dy3,dy3
    std         dy2,[dy+16]
    add         dx,32,dx
    add         dy,32,dy
    addcc       n,-4,n
    bg          loop_top
    std         dy3,[dy-8]! same as dy+24
```

Figure 7. LINPACK benchmark code

System Considerations

INTERFACE TO IU AND MEMORY

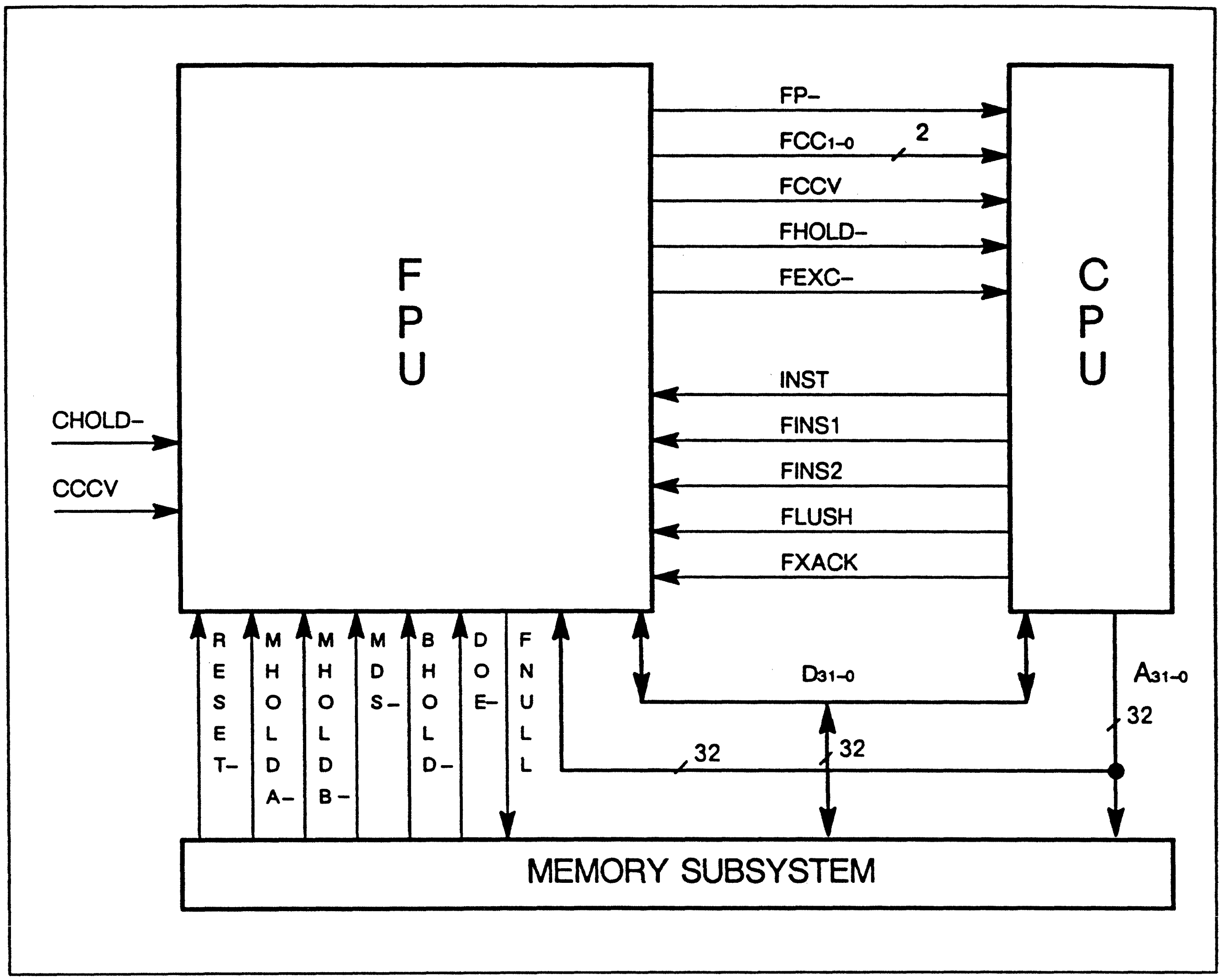


Figure 8. Interface to IU and memory

System Considerations, continued

INSTRUCTION OPERATION

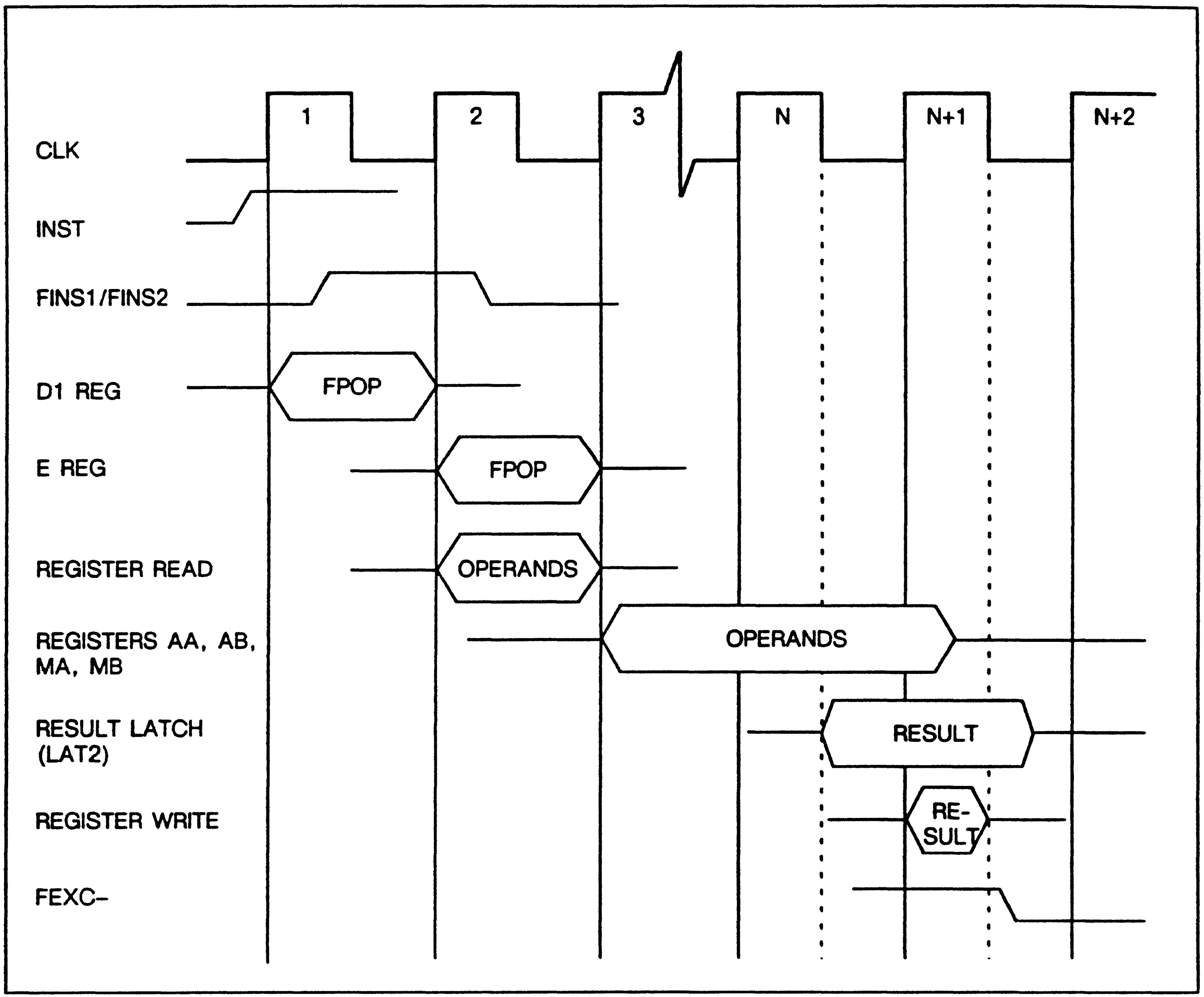


Figure 9. Instruction operation

Specifications

Supply voltage	–0.5 to 7.0 V
Input voltage	–0.5V to VCC
Output voltage	–0.5V to VCC
Operating temperature range (TCASE)	0° to 85° C
Storage temperature range	–65° C to 150° C
Lead temperature (10 seconds)	300° C
Junction temperature	155° C

Figure 10. Absolute maximum ratings

PARAMETER		MIN	MAX	UNIT
V_{CC}	Supply voltage	4.75	5.25	V
I_{OH}	High-level output current		-1.0	mA
I_{OL}	Low-level output current		4.0	mA
T_{CASE}	Operating case temperature	0	85	°C

Figure 11. Operating conditions

DC SPECIFICATIONS

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
V_{IH}	High–level input voltage	V_{CC} = MIN	2.1		V
V_{IL}	Low–level input voltage	V_{CC} = MIN		0.8	V
V_{IHC}	High–level input voltage	V_{CC} = MIN	2.4		V
V_{ILC}	Low–level input voltage	V_{CC} = MIN		0.8	V
V_{OH}	High–level output voltage	V_{CC} = MIN, I_{OH} = MAX	2.4		V
V_{OL}	Low–level output voltage	V_{CC} = MIN, I_{OL} = MAX		0.4	V
I_{LI}	Input leakage current	V_{CC} = MAX, V_{IN} = 0 to V_{CC}		±10	µA
I_{LO}	Output leakage current (output disabled)	V_{CC} = MAX, V_{IN} = 0 or V_{CC}		±10	µA
C_{IN}	Input capacitance*	V_{CC} = MAX, V_{IN} = 0 to Vcc		15	pF
C_{OUT}	Output capacitance*	V_{CC} = MAX, V_{OUT} = 0 to Vcc		20	pF
C_{CLK}	Clock Input capacitance*	Vcc = MAX, V_{IN} = 0 to Vcc		25	pF
C_{DOE-}	DOE- Input capacitance*	Vcc = MAX, V_{IN} = 0 to Vcc		30	pF
I_{CC}	Supply current	V_{CC}= MAX, T_{CY} = MIN; TTL inputs			mA
* Guaranteed, but not tested					

Figure 12. DC specifications

Specifications, continued

AC SPECIFICATIONS AND TIMING DIAGRAMS

PARAMETER		Min/Max	Reference (Note 1)	25 MHz	33 MHz	40 MHz	UNITS
TCY	Clock Cycle	MIN		40	30	25	ns
TCH	Clock High	MIN		18	13	11	ns
TCL	Clock Low	MIN		18	13	11	ns
TCR	Clock Rise Time (Note 2)	MAX		1	1	1	V/ns
TCF	Clock Fall Time (Note 2)	MAX		1	1	1	V/ns
T1	A bus (Address) Setup	MIN	CLK+	3	3	2	ns
T2	A bus (Address) Hold	MIN	CLK+	6	6	6	ns
T3	D bus (Data) Load Setup	MIN	CLK+	3	2	2	ns
T4	D bus (Data) Load Hold	MIN	CLK+	5	5	4	ns
T5	FINS1/FINS2 Setup	MIN	CLK+	9	9	7	ns
T6	FINS1/FINS2 Hold	MIN	CLK+	2.5	2.5	2.5	ns
T7	INST Setup	MIN	CLK+	16	12	9	ns
T8	INST Hold	MIN	CLK+	2	2	2	ns
T9	FXACK Setup	MIN	CLK+	16	12	9	ns
T10	FXACK Hold	MIN	CLK+	2	2	2	ns
T11	FLUSH Setup	MIN	CLK+	21	14	11	ns
T12	FLUSH Hold	MIN	CLK+	2	2	2	ns
T13	RESET- Setup	MIN	CLK+	15	10	8	ns
T14	RESET- Hold	MIN	CLK+	3	3	2	ns
T15	MHOLDA- Setup (Note 3)	MIN	CLK-	7	4	3	ns
T16	MHOLDA- Hold (Note 3)	MIN	CLK-	6	5	4.5	ns
T17	MDS- Setup	MIN	CLK-	5	4	3	ns
T18	MDS- Hold	MIN	CLK-	6	5	4.5	ns
T19	FHOLD Delay	MAX	CLK-	29	23	19	ns
T20	FHOLD Valid	MIN	CLK-	8	6	5.5	ns
T21	FHOLD Delay	MAX	FINS1/FINS2	16	15	12	ns
T22	FHOLD Delay	MAX	FLUSH	28	20	16	ns
T23	FHOLD Delay	MAX	MHOLDA- (Note 3)	36	27	22	ns
T24	FCCV Delay	MAX	CLK-	29	23	19	ns
T25	FCCV Valid	MIN	CLK-	8	6	5.5	ns
T26	FCCV Delay	MAX	FLUSH	28	20	16	ns
T27	FCCV Delay	MAX	MHOLDA- (Note 3)	36	27	22	ns
T28	FCC (1-0) Delay	MAX	CLK+	26	19	17	ns
T29	FCC (1-0) Valid	MIN	CLK+	5	4	3	ns
T30	FEXC- Delay	MAX	CLK+	26	19	17	ns
T31	FEXC- Valid	MIN	CLK+	5	4	3	ns
T32	FNULL Delay	MAX	CLK+	20	13	11	ns
T33	FNULL Valid	MIN	CLK+	3	3	3	ns
T34	D bus (Data) Store Delay	MAX	CLK-	20	15	13	ns
T35	D bus (Data) Store Valid	MIN	CLK-	4	4	4	ns
T36	D bus Turn-off Time (Note 2)	MAX	FLUSH (Note 4)	31	22	18	ns
T37	D Bus Store Valid	MIN	FLUSH (Note 4)	0	0	0	ns
T38	D bus Turn-off Time (Note 2)	MAX	DOE-	15	11	9	ns
T39	D bus Turn-on Time (Note 2)	MAX	DOE-	15	11	9	ns
T40	D bus Store Valid	MIN	DOE-	0	0	0	ns

Note 1. "CLK+" means with respect to a rising edge, "CLK-" means with respect to a falling edge of the clock
Note 2. This parameter is guaranteed but not tested
Note 3. This specification applies also to MHOLDB-, BHOLD-, CHOLD-, and CCCV signals
Note 4. When changing state, FLUSH may switch only once in any given cycle. This means that FLUSH has to come out of a register triggered on a positive clock edge.

Figure 13. AC specifications

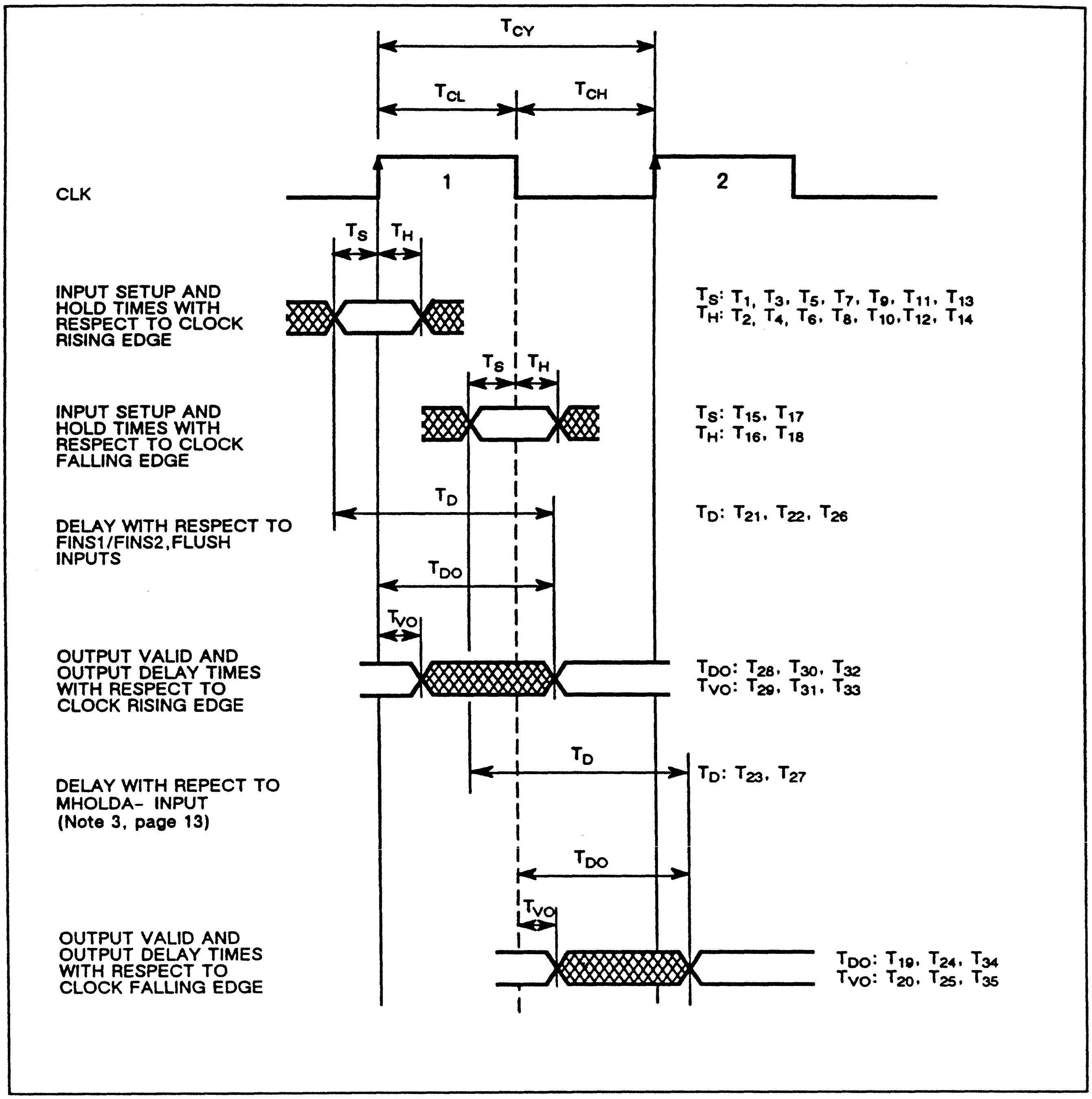

T_{CY}
T_{CL}
T_{CH}
CLK
1
2
T_S
T_H
INPUT SETUP AND HOLD TIMES WITH RESPECT TO CLOCK RISING EDGE
T_S: T_1, T_3, T_5, T_7, T_9, T_{11}, T_{13}
T_H: T_2, T_4, T_6, T_8, T_{10}, T_{12}, T_{14}
T_S
T_H
INPUT SETUP AND HOLD TIMES WITH RESPECT TO CLOCK FALLING EDGE
T_S: T_{15}, T_{17}
T_H: T_{16}, T_{18}
T_D
DELAY WITH RESPECT TO FINS1/FINS2,FLUSH INPUTS
T_D: T_{21}, T_{22}, T_{26}
T_{DO}
T_{VO}
OUTPUT VALID AND OUTPUT DELAY TIMES WITH RESPECT TO CLOCK RISING EDGE
T_{DO}: T_{28}, T_{30}, T_{32}
T_{VO}: T_{29}, T_{31}, T_{33}
T_D
DELAY WITH REPECT TO MHOLDA- INPUT (Note 3, page 13)
T_D: T_{23}, T_{27}
T_{DO}
T_{VO}
OUTPUT VALID AND OUTPUT DELAY TIMES WITH RESPECT TO CLOCK FALLING EDGE
T_{DO}: T_{19}, T_{24}, T_{34}
T_{VO}: T_{20}, T_{25}, T_{35}

Figure 14. Timing diagrams

Specifications, continued

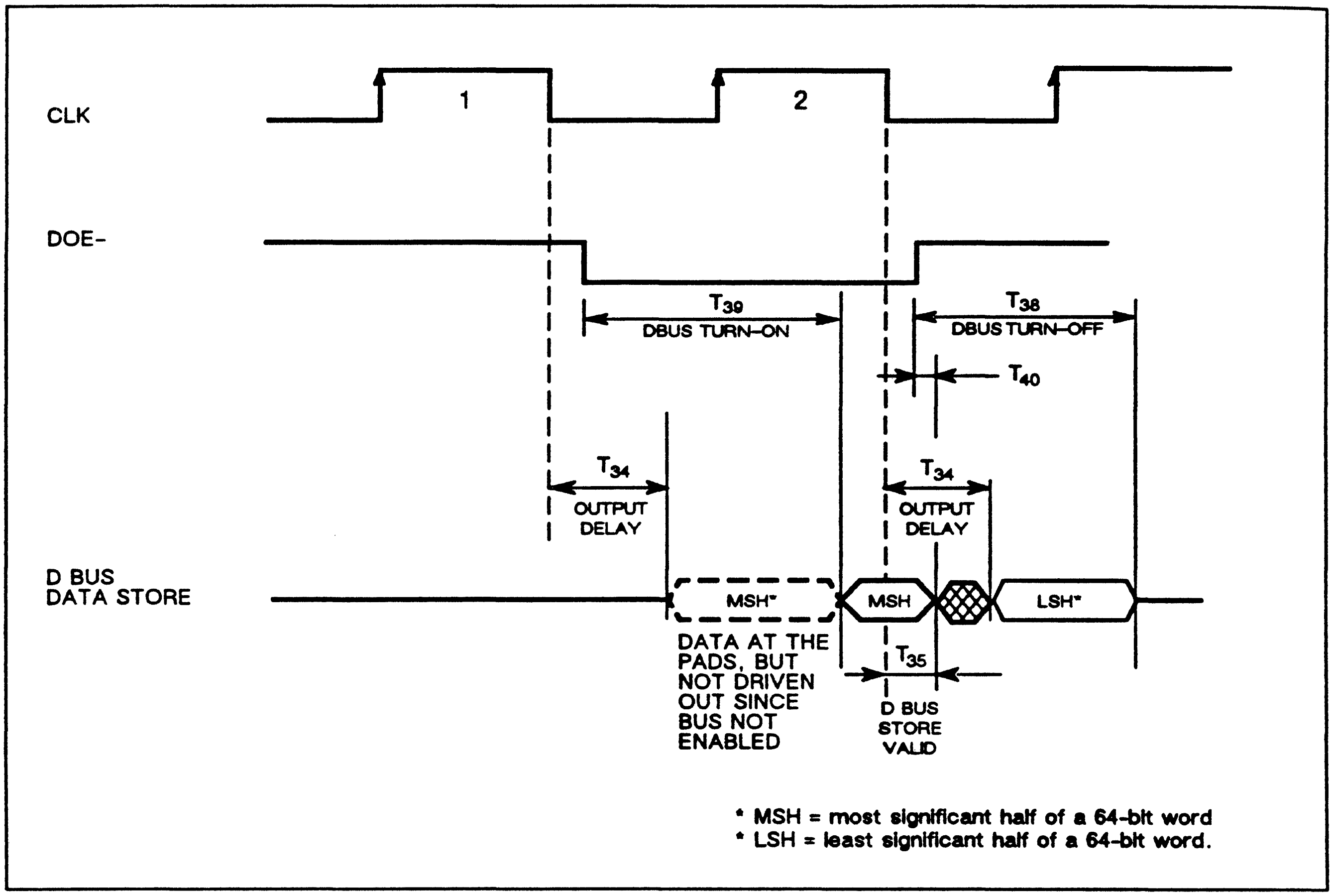

CLK
1
2
DOE-
T39
DBUS TURN-ON
T38
DBUS TURN-OFF
T40
T34
OUTPUT DELAY
T34
OUTPUT DELAY
D BUS DATA STORE
MSH*
MSH
LSH*
DATA AT THE PADS, BUT NOT DRIVEN OUT SINCE BUS NOT ENABLED
T35
D BUS STORE VALID
* MSH = most significant half of a 64-bit word
* LSH = least significant half of a 64-bit word.

Figure 15. Asynchronous data bus store timing

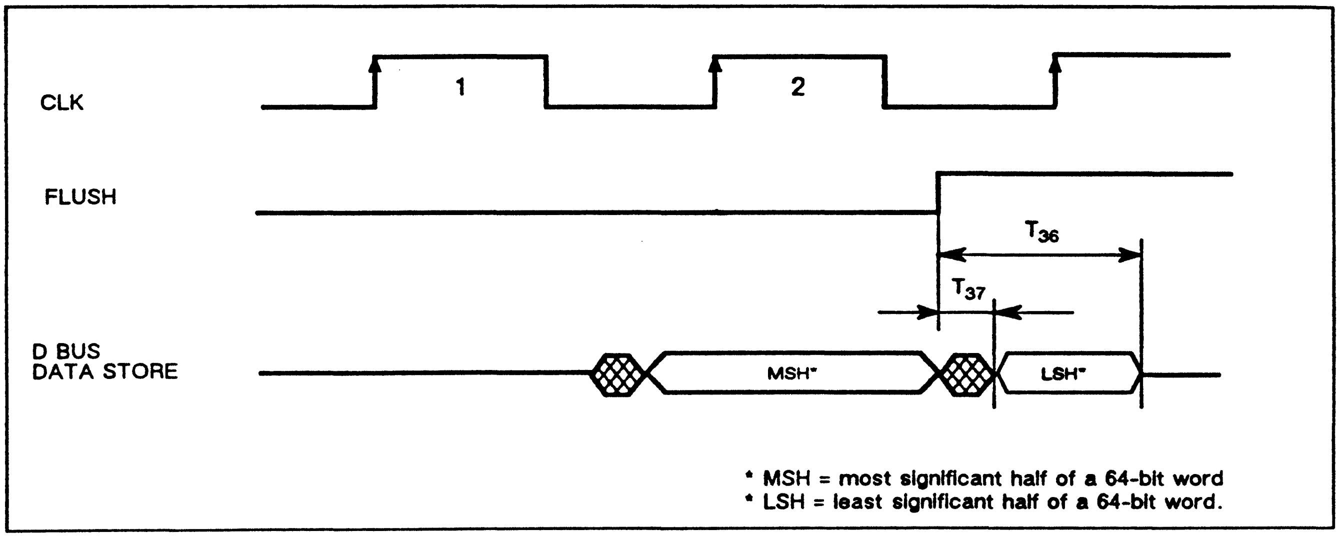

CLK
1
2
FLUSH
T36
T37
D BUS DATA STORE
MSH*
LSH*
* MSH = most significant half of a 64-bit word
* LSH = least significant half of a 64-bit word.

Figure 16. The effect of FLUSH on data bus store timing

Specifications, continued

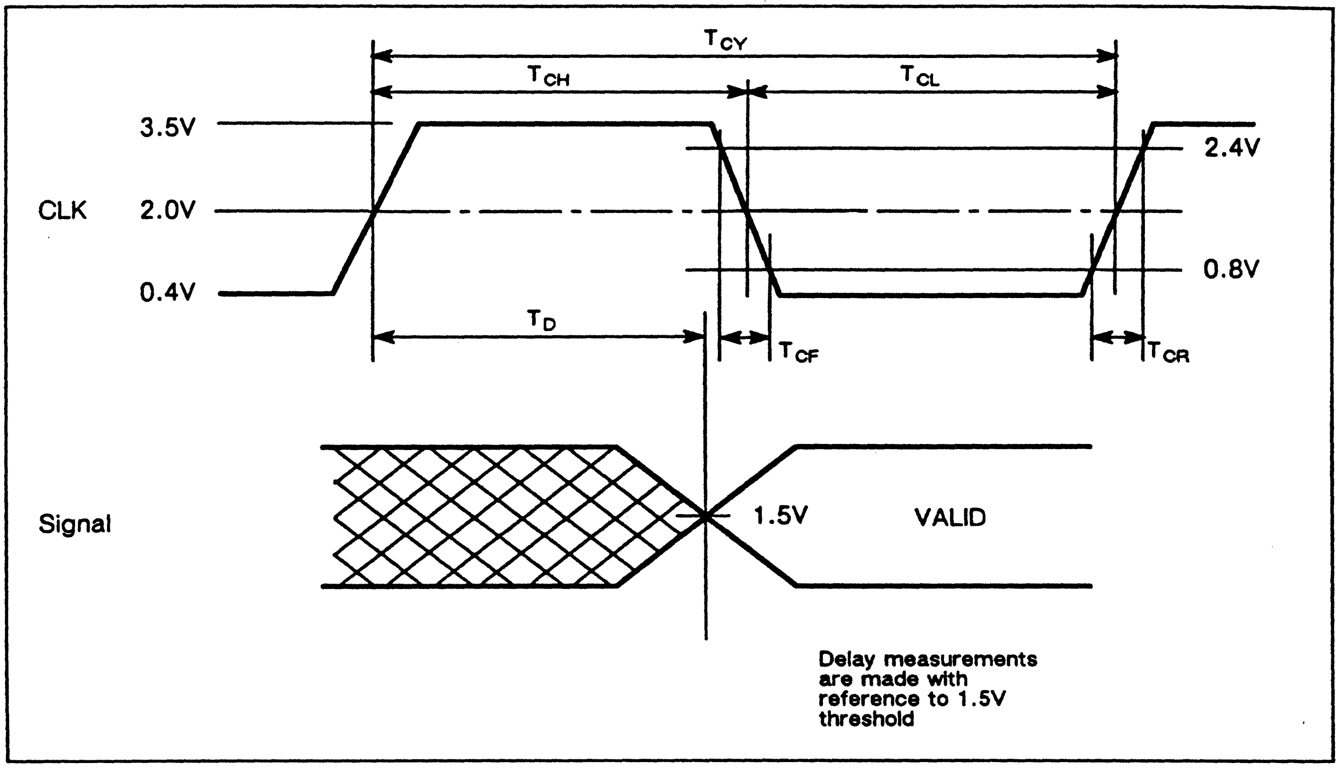


Figure 17. Reference levels in delay measurements

Specifications, continued

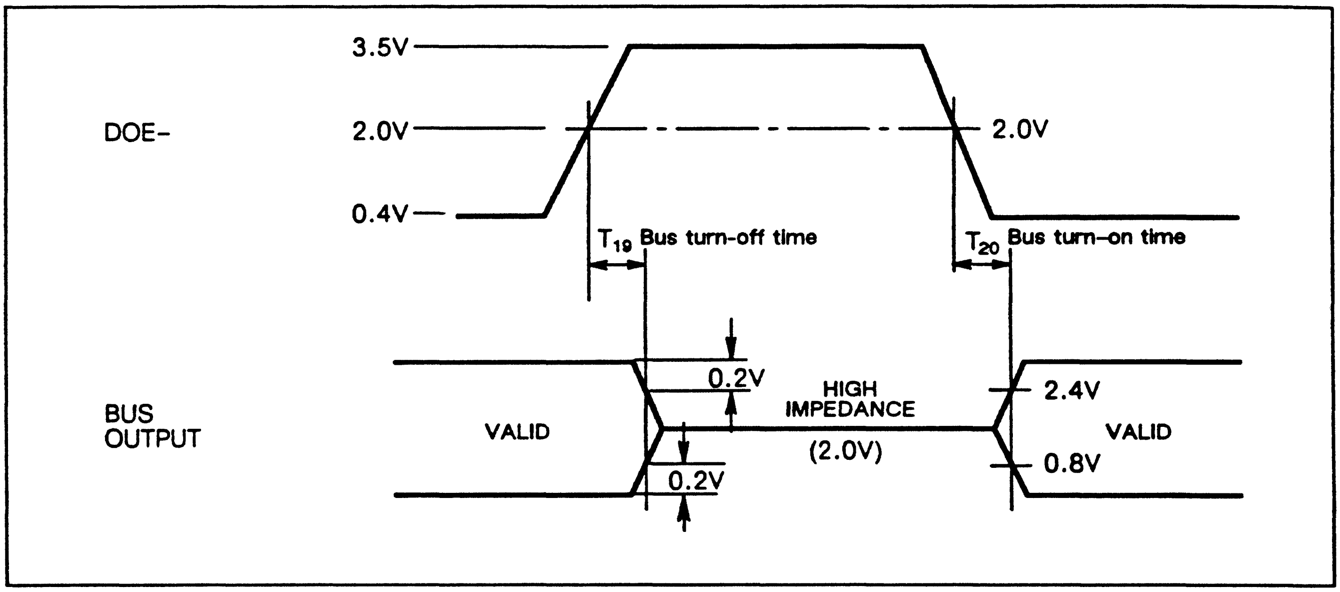


Figure 18. Tri-state timing

I/O CHARACTERISTICS

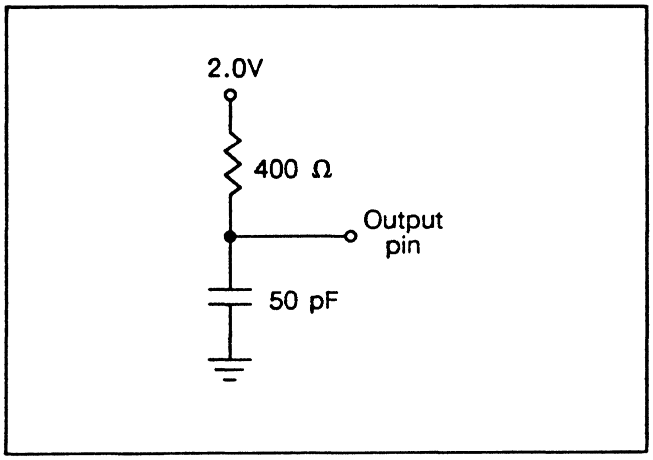
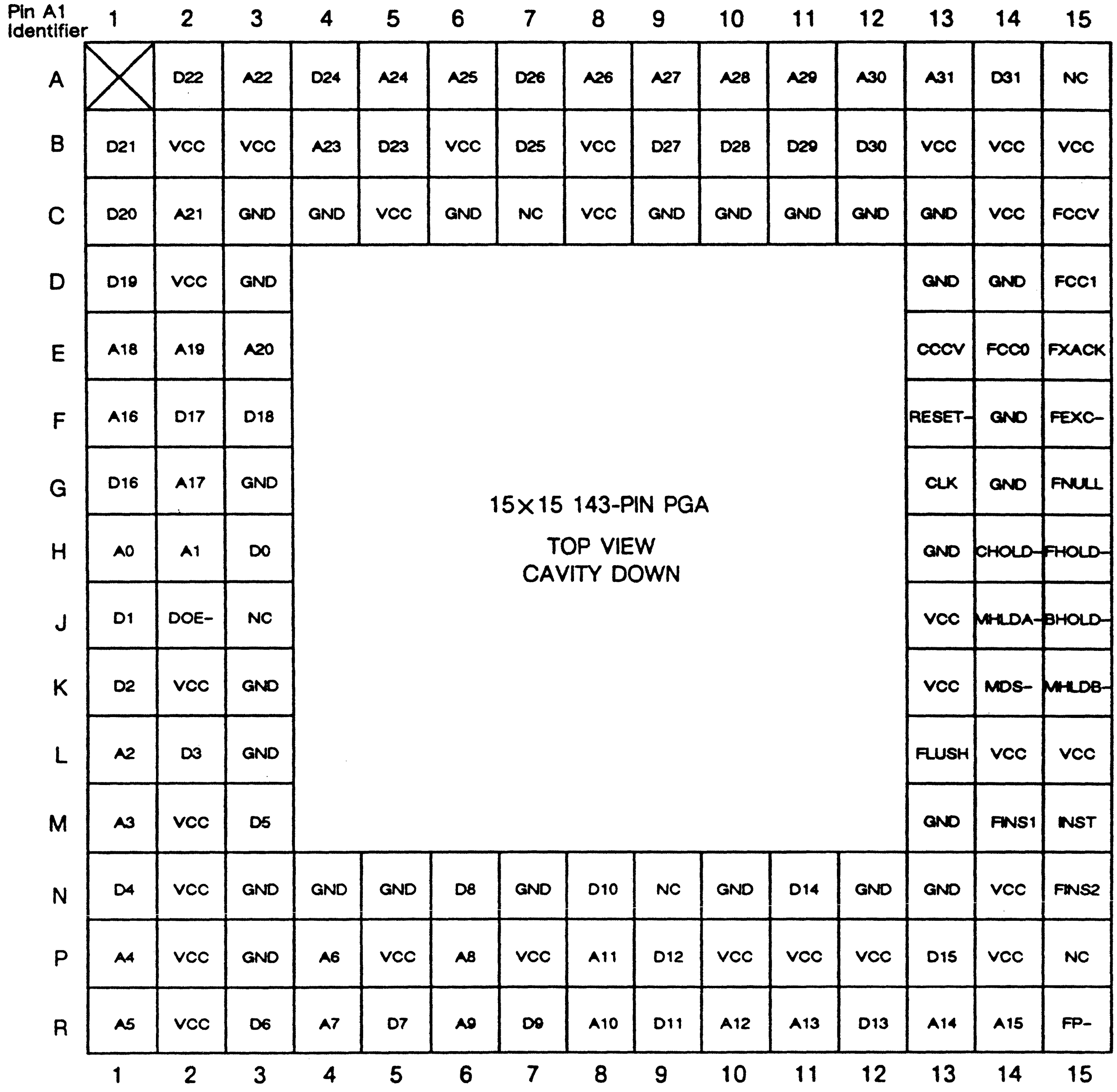


Figure 19. AC test load

Pin Configuration



Pin A1 Identifier	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
A	☒	D22	A22	D24	A24	A25	D26	A26	A27	A28	A29	A30	A31	D31	NC	A
B	D21	VCC	VCC	A23	D23	VCC	D25	VCC	D27	D28	D29	D30	VCC	VCC	VCC	B
C	D20	A21	GND	GND	VCC	GND	NC	VCC	GND	GND	GND	GND	GND	VCC	FCCV	C
D	D19	VCC	GND										GND	GND	FCC1	D
E	A18	A19	A20										CCCV	FCC0	FXACK	E
F	A16	D17	D18										RESET-	GND	FEXC-	F
G	D16	A17	GND										CLK	GND	FNULL	G
H	A0	A1	D0										GND	CHOLD-	FHOLD-	H
J	D1	DOE-	NC										VCC	MHLDA-	BHOLD-	J
K	D2	VCC	GND										VCC	MDS-	MHLDB-	K
L	A2	D3	GND										FLUSH	VCC	VCC	L
M	A3	VCC	D5										GND	FINS1	INST	M
N	D4	VCC	GND	GND	GND	D8	GND	D10	NC	GND	D14	GND	GND	VCC	FINS2	N
P	A4	VCC	GND	A6	VCC	A8	VCC	A11	D12	VCC	VCC	VCC	D15	VCC	NC	P
R	A5	VCC	D6	A7	D7	A9	D9	A10	D11	A12	A13	D13	A14	A15	FP-	R
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

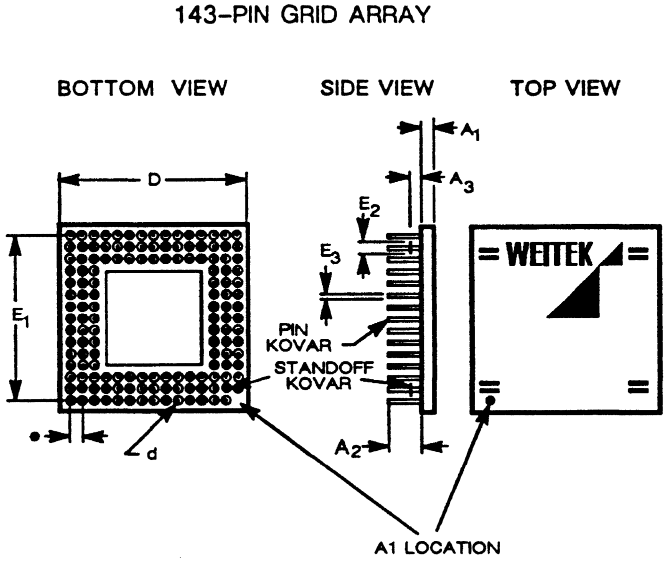
15×15 143-PIN PGA

TOP VIEW
CAVITY DOWN

Note: NC = not connected; pins so marked must be left unconnected.
There is no pin at A1. A1 is a locator hole.

Figure 20.

Physical Dimensions



Symbol	DIMENSIONS	
	INCHES	MM
A1	0.100 ± 0.010	2.54 ± .20
A2	0.180 typ.	4.57 typ.
A3	0.050 typ.	1.27 typ.
D	1.575 sq. ± 0.016	40.0 ± 0.41
E1	1.400 sq. ± 0.012	35.56 ± 0.30
E2	0.050 dia. typ.	1.27 dia. typ.
E3	0.018 ± 0.002	.46 ± 0.05
d	0.065 dia. typ.	1.65 dia. typ.
e	0.100 typ.	2.54 typ.

Ordering Information

Package Type	Frequency	Case Temperature Range	Order Number
143-pin PGA	25 MHz	0–85°C	3171–025–GCD
143-pin PGA	33 MHz	0–85°C	3171–033–GCD
143-pin PGA	40 MHz	0–85°C	3171–040–GCD

WEITEK

Headquarters

WEITEK Corporation
1060 East Arques
Sunnyvale, CA 94086
TEL (408) 738-8400
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185

WEITEK U.S.A.

WEITEK Corporation
1060 East Arques
Sunnyvale, CA 94086
TEL (408) 738-8400
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185

Corporate Place IV
111 South Bedford St.
Suite 200
Burlington, MA 01803
TEL (617) 229-8080
FAX (617) 229-4902

WEITEK Europe

Greyhound House
23/24 George St.
Richmond, Surrey
England TW9 1JY
TEL (011) 441-948-8608
TELEX 928940 RICHBI G
FAX (011) 441-940-6208

WEITEK Japan

4-8-1 Tsuchihashi
Miyamae-Ku
Kawasaki, Kanagawa-Pre
213 Japan
TEL 044-852-1135
FAX 044-877-4268