WEITEK

XL 3232 32-BIT GRAPHICS FLOATING POINT COMPUTATION UNIT

PRELIMINARY DATA
August 1988

The WEITEK XL 3232 single-chip graphics floating point unit offers a full instruction set including multiply, multiply/accumulate, add, subtract, type conversion, and divide look-up operations. Efficient design and architecture, combined with CMOS technology, provide up to 25 MFLOPS of performance at very low power. Its on-chip register file and MAC architecture make the device useful for Bezier evaluations, matrix transforms, and other graphics operations.

Related products: XL-8236 32-bit raster code sequencer, XL-8237 32-bit raster image processor

WEITEK
TIME: 2 Pm DATE: 8/1
PRELIMINARY
TRANSMITTED BY: Julie Cater

Contents

XL-3232 Graphics Floating Point Computation Unit Data Sheet
August 1988

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

Printed in the United States of America
90 89 88 7 6 5 4 3 2 1

XL-3232
32-BIT GRAPHICS FLOATING POINT COMPUTATION UNIT

PRELIMINARY DATA

August 1988

Features

32-BIT FLOATING-POINT PROCESSOR

Single-precision floating-point multiplier/ALU

Four-port 32×32 register file for local variable and matrix element storage

Low power, high integration CMOS

FULL FUNCTION

Add, subtract, multiply, multiply/accumulate

Divide look-up table

Type conversion to and from two's complement integer

Three-address (*rc* := *ra* + *rb*) architecture

HIGH PERFORMANCE

20 to 60 page per minute PostScript® print times

Up to 25 MFLOPS throughput (1 MAC/cycle)

Low latency (three-cycle register-to-register operations)

Description

The XL-3232 is a single-precision floating-point computation unit. It includes a pipelined multiplier/accumulator and a four-port register file with thirty-two 32-bit registers.

The XL-3232 is suited to a wide range of systems that need high-performance image and graphics processing.

The XL-3232 has a single bi-directional 32-bit input/output port.

The XL-3232 is used with the WEITEK XL-8236 raster code sequencer (RCS) and XL-8237 raster image processor (RIP) to create a fast, general-purpose numeric processor, the XL-8232. Full development system support—including a C compiler and a PostScript-compatible interpreter—is available for the XL-8200 Series of processors.

The XL-3232 is a low-power CMOS device which is available in a standard 144-pin ceramic PGA (pin grid array) package.

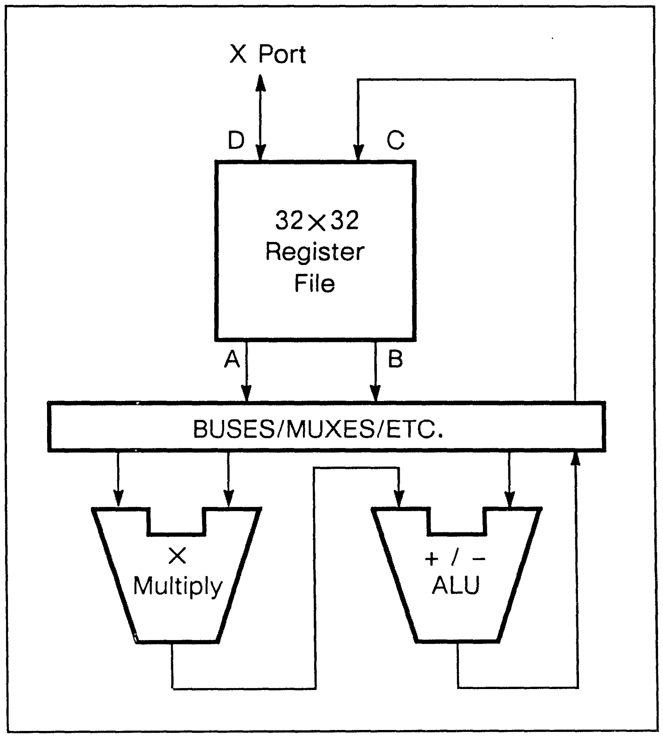


Figure 1. XL-3232 core functions

Architecture

MULTIPLIER/ACCUMULATOR

The core of the XL-3232 is the *multiplier/accumulator pipeline*. Its first stage can multiply two operands together. The next stage can add or subtract another operand. Finally, the result is rounded and returned to a register and/or output port.

Multiply, add, subtract, and multiply/accumulate operations are performed in the multiplier/accumulator. They all operate on data that conforms to the IEEE single-precision floating-point format.

Each operation takes three cycles, but a new operation can be started on every cycle because the multiplier/accumulator is pipelined. Three independent operations may be at different stages in their execution at any time.

Rounding, conversion between floating-point and two's complement integer formats, and other miscellaneous functions are performed in the accumulator.

REGISTER FILE

Operands and results of the multiplier/accumulator may be stored in the four-port *register file*. This file contains thirty-two registers, each of which may store a 32-bit value.

The four ports allow the register file to supply two operands to the multiplier/accumulator, store its result back to a register, and perform an input/output transfer—all in the same cycle.

INPUT/OUTPUT PORT

The *external I/O port* is 32 bits wide. It can transfer a data value on every cycle.

The XL-3232 has one bi-directional external port; the X port. It can load and store data to and from the register file, and it can transfer data directly to and from the multiplier/accumulator.

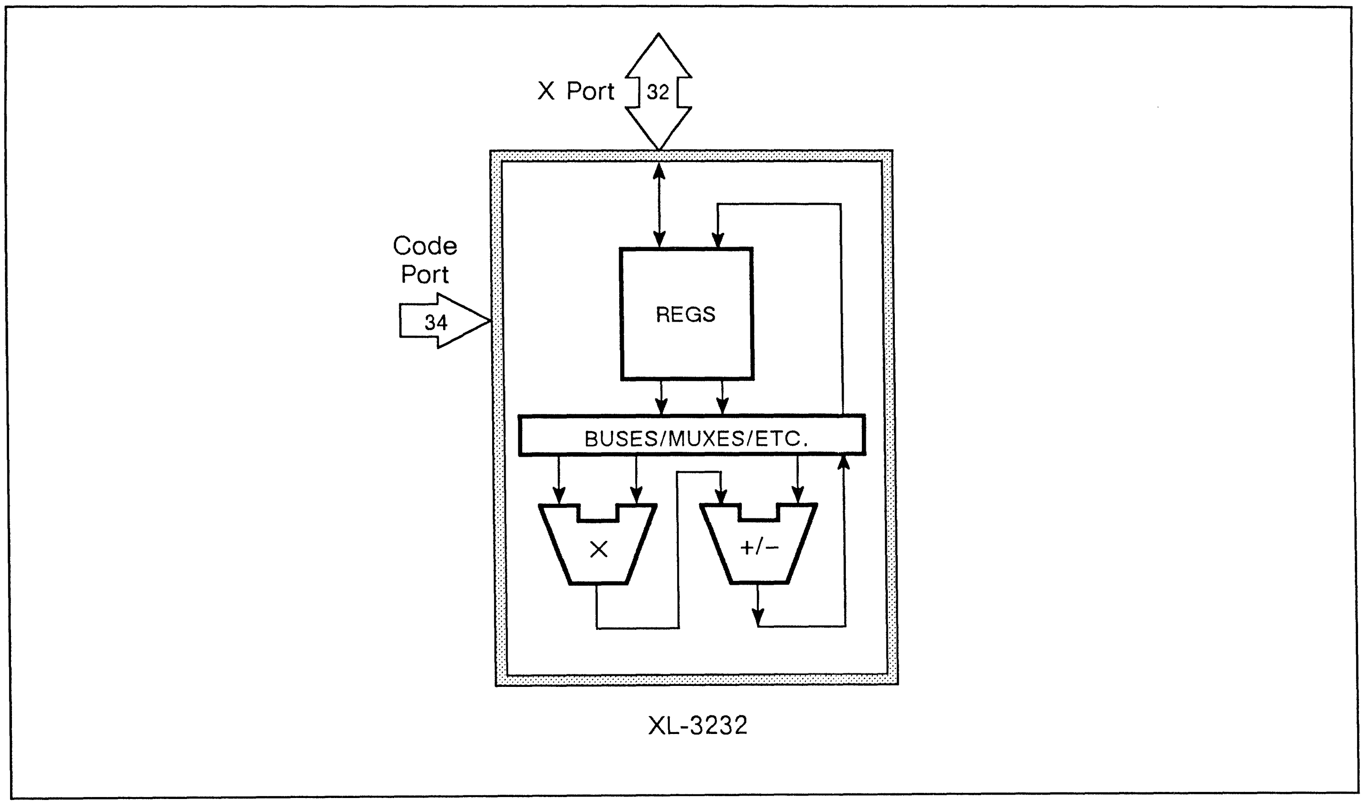


Figure 2. XL-3232 I/O

Architecture, continued

TEMPORARY REGISTERS

Three 32-bit *temporary registers* are provided to store intermediate results. They make it possible to perform operations of the form

$x = x \pm (y \times z)$

in a single cycle.

DIVIDE LOOK-UP TABLE

Support for divide operations is provided by an on-chip *look-up table*. It returns an approximation for the inverse of a value which is then refined by iterative multiply/accumulate operations. Division is accomplished by multiplying the dividend by the inverse of the divisor. This complete divide operation takes eighteen cycles; other operations may be interleaved without a performance penalty.

INSTRUCTIONS

An instruction is latched into the code port on every cycle. An instruction specifies operand sources, a result destination and all of the steps that will create this result during the next three cycles. Condition codes and exceptions may be generated by each operation as the result is written back to the register file.

Four five-bit fields provide addresses for the register file. They each select a source or destination for one of the register ports. The three-bit function field specifies the type of multiplier/accumulator operation. The two-bit I/O control field directs data transfer at the external X port. Other fields select the route taken by the data during the the operation.

A mode register controls data routing options that rarely change, and selects between a number of I/O timing options.

XL-8200 SERIES COMPATIBILITY

The XL-3232 is used with the WEITEK XL-8236 *raster code sequencer* (RCS) and XL-8237 *raster image processor* (RIP) to create the XL-8232 processor.

The XL-3232 *graphics floating-point unit* (FPU) shares a 64-bit instruction word with the RIP and RCS. The RIP and FPU also share the 32-bit-wide data bus.

The XL-3232 responds to the NEUT- and STALL- signals used to control branching and "wait states" within the XL-8232. It communicates its status to the RCS with the floating-point condition (FPCN) and exception (FPEX) lines.

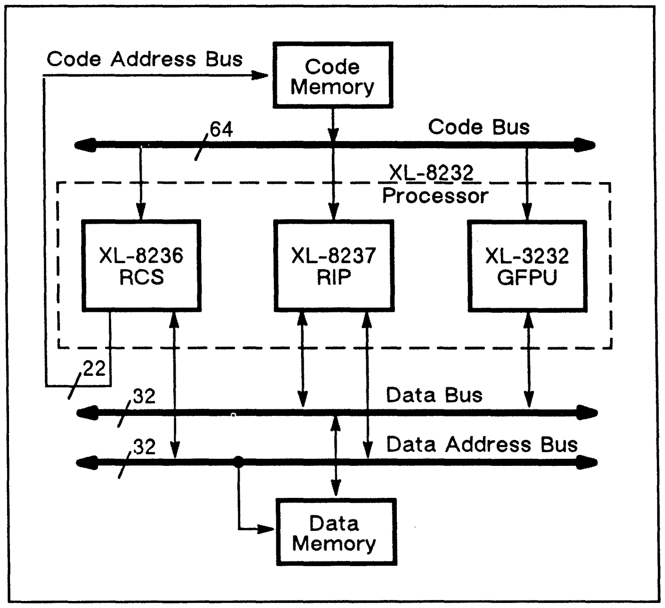


Figure 3. XL-8232 block diagram

Signal Description

X PORT

The 32-bit $X_{31..0}$ port is a bi-directional data bus. Input data is sampled on the rising edge of CLK (or, if double-pump mode is enabled, both on the rising and falling edges of CLK). Data transfers are controlled by the $IOCt_{1..0}$ field in the instruction word. The X port may be set to a high impedance state by the OEX– signal. Active high.

C PORT

The 34-bit $C_{33..0}$ port is used as a code input bus. Instructions are latched the rising edge of CLK. Active high.

OEX–

X port output enable input. OEX– asynchronously disables the X port when high. Active low.

FPEX–

Floating-point exception output. FPEX signals the occurrence of an enabled exception (overflow).

FPCN

Floating-point condition output. FPCN signals the occurrence of a condition as specified in the $Encn_{1..0}$ field of an instruction. Active high.

ZERO

Zero condition output. Indicates that the result of an operation is exactly equal to zero. Controlled by the $Encn_{1..0}$ field of an instruction. Active high.

NEUT–

Neutralize input. Cancels the effect of the current instruction. Typically used during delayed branches and interrupt response routines (see page 27). Latched on the cycle following the instruction to be cancelled. Active low.

STALL–

Stall input. Cancels the effect of the next instruction. Typically used as a "not ready" line from the code memory (see page 28). Latched on the same cycle as the potentially invalid instruction. Active low.

CLK

Clock input. TTL compatible.

VDD

All VDD pins must be connected to 5.0V.

GND

All GND pins must be connected to system ground.

Note: Signals denoted by "–" are active low.

Block Diagram

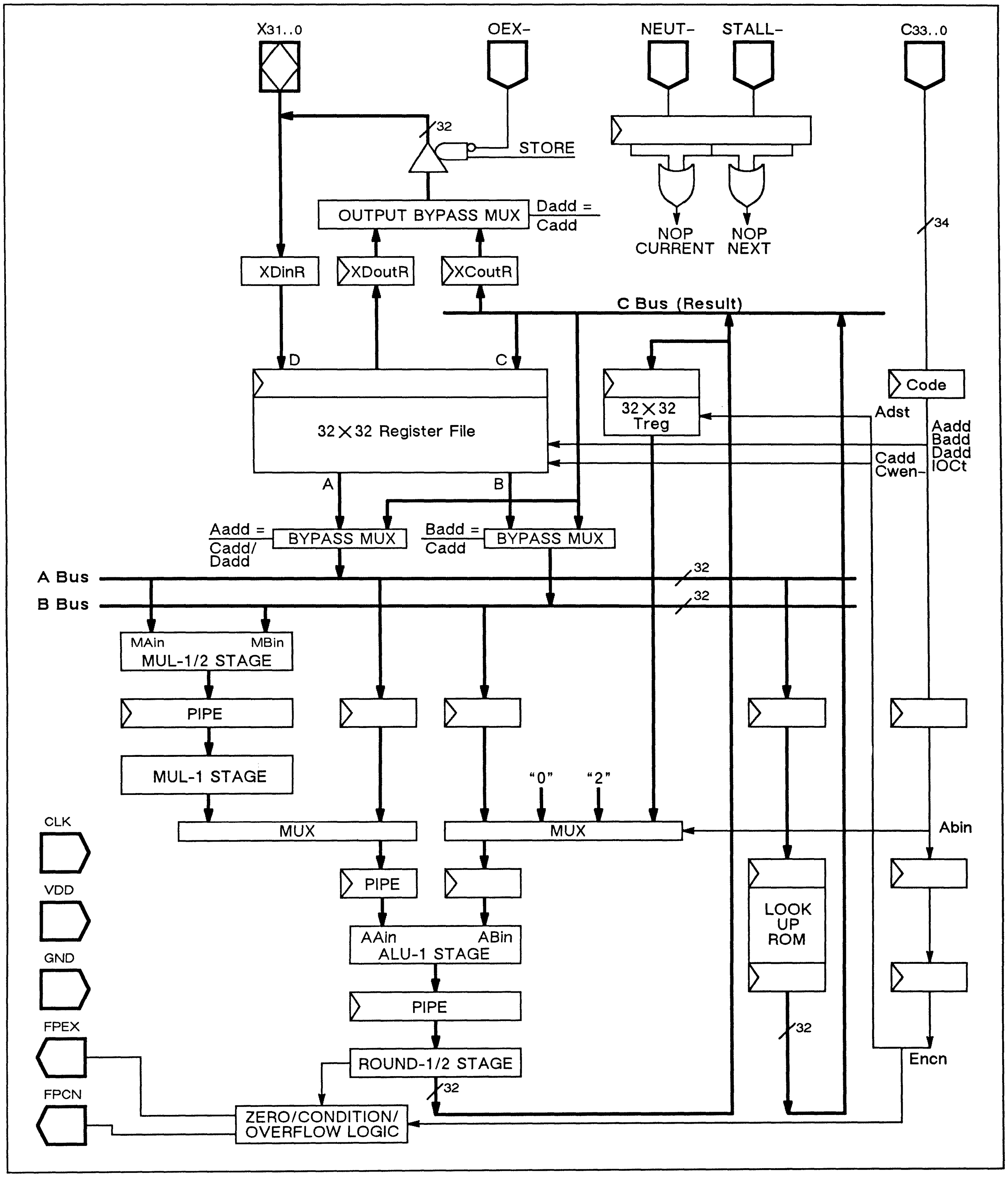


Figure 4. XL-3232 block diagram

XL-8232 Architecture

WEITEK XL-8200 SERIES

The WEITEK XL-8200 Series is a family of two VLSI processors: the XL-8200, a high-speed 32-bit integer processor; and the XL-8232, a single-precision floating-point processor.

These processors give the performance of RISC processors, and are supported by a full complement of development tools. These include C and FORTRAN 77 compilers and an assembler. A development system offers both hardware and software simulators with debugging facilities. The programmer remains free to create custom assembly-language routines for peak performance.

This data sheet is dedicated to the XL-8232 single-precision floating-point processor. Further information may be found in the *XL-8200 Overview*, the *XL-8200 Designer's Binder*, the *XL-8236 Data Sheet,* and the *XL-8237 Data Sheet.*

The XL-8232 processor consists of three interconnected VLSI components:

- XL-8236 raster code sequencer (RCS)
- XL-8237 raster image processor (RIP)
- XL-3232 graphics floating-point unit (FPU)

Each of these components is manufactured in high-density, low-power CMOS. They are delivered in 144-pin PGA packages.

The XL-8200 Series simplifies system design. Zero-glue interfacing is provided by a small number of dedicated signals that communicate state information between the components. These signals and the system buses need only be connected as shown in figure 8 in order to create the XL-8232. The purpose of each interconnection is described in detail below.

BUSES

Four high-bandwidth system buses are provided by the XL-8232:

1. Code bus.

   The 64-bit code bus feeds the code input ports of the RCS, RIP and FPU. The RCS and RIP share 32 of the 64-bits; this half of the code word directs program control operations, address generation, loads and stores, and integer arithmetic. The remainder of the code word directs floating-point operation.

2. Data bus.

   The 32-bit data bus is shared by the RIP and FPU. It allows bytes, 32-bit integers and 32-bit floating-point numbers to be transferred between the processing units and data memory.

3. Code Address bus.

   The 22-bit code address bus carries the address of the next instruction from the RCS to the code memory. A word address allows up to four megawords of 64-bit wide code memory. (This bus does not connect to the XL-3232 FPU.)

4. Data Address bus.

   A 32-bit data address bus carries the address of the next data read or write. The address is generated by the RIP and the data may be transferred to or from the RIP or FPU as required. A byte address allows up to 4 Gbytes of 32-bit wide data memory. Support for accessing bytes, half-words, and words is provided by the RIP. (This bus does not connect to the XL-3232 FPU.)

The XL-3232 hooks directly to the code and data buses alongside the other components of the XL-8232. The code word is sampled by all three components simultaneously. The data bus is driven or sampled at the same time in the cycle no matter which component is transferring information.

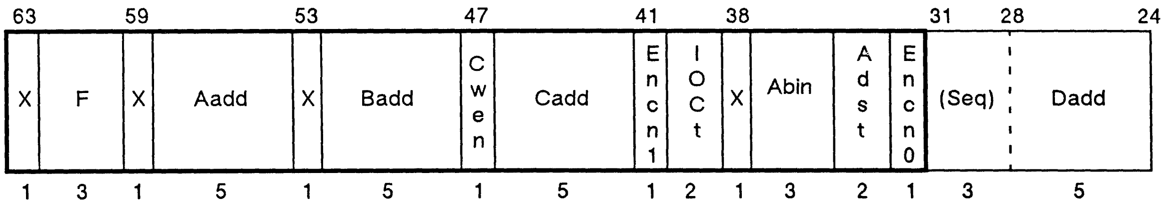
The code and data memory systems can be implemented with ROM, SRAM, DRAM, or static column DRAM. Both code and data caches can be used with the XL-8232 chip set.

INSTRUCTION FORMAT

The XL-8232 has a 64-bit instruction word. The bits that are directed to the XL-3232 are shown in figure 5.

The lower 32 bits of the instruction word are shared by the RIP and the RCS. Bits 0–23 normally define the RIP operation. Bits 24–31 define the instruction flow control performed by the RCS. Five of these control bits, 24–28, are also used as the floating-point register address (Dadd) when floating-point load and store operations are performed. This saves on code bits and insures that the FPU and RIP never compete for the data bus.

XL-8232 Architecture, continued



1. Dashed lines indicate bits in the sequencer field.
2. Bits marked with an "X" are reserved bits.
3. The meaning of each bit field is described in the body of this data sheet.

Figure 5. XL-3232 signal assignments in the XL-8232 code word

LOAD/STORE MODEL

The XL-8200 Series has a consistent load/store model regardless of processor configuration. Each processing unit has its own register file. Register moves between the RIP and FPU must be made through the data memory. Each of these register files is multi-ported and each register may be the operand source or result destination of any instruction implemented by the unit. When an instruction takes more than one cycle to execute, the registers that supply its operands and receive its result must not be modified until it has been completed.

Transactions between the register files and data memory are performed with coprocessor load/store instructions. The only restriction on loads and stores is that the operands of an operation be loaded before it is executed and that it shall have completed before its result is stored. These simple restraints allow the parallelizer considerable freedom to optimize register usage and I/O transactions.

An example of the normal sequence of operation is given below. This example leaves several free cycles in which other loads, stores, and calculations could be performed in parallel.

```
.
.
addr .ra
fload .fx
fabs .fx, .fy
nop
addr .rb
fstore .fy
.
.
```

Figure 6. Code example

XL-8232 Architecture, continued

The fload and fstore operations have the same timing as the RIP's load and store operations. For loads, the address is presented on the AD bus at the beginning of a cycle; the data is expected to be available on the D bus by the end of that cycle. For stores, the address is presented on the AD bus at the beginning of a cycle and the data is driven onto the D bus during the *next* cycle.

Because the addr and fload instructions can be executed in parallel, they may be pipelined to support contiguous load operations (one per cycle). If the external data memory system provides a write buffer, then stores can be pipelined in similar fashion.

If, however, loads and stores are to be interleaved, each store must be allowed two cycles; the latter cycle must contain an I/O nop (that is, it must not contain a load or store instruction). This has minimal impact on overall performance because loads usually outnumber stores; and the parallelizer can organize I/O transfers efficiently. If the code constraints covered in the body of this data sheet are followed or if WEITEK software tools are used, then this load/store model will be obeyed.

MODES

The XL-3232 Mode Register must be initialized to the values given in figure 7 when used in the XL-8232 processor. The resulting programming model is illustrated in figure 4. Optional selections are:

1. The fix and float range test may be enabled or disabled as required.
2. Overflows may be enabled or disabled as required.

MODE BIT	LOGIC VALUE	OPTIONAL?	DESCRIPTION
M0	1	NO	Internal Bypass Mode (Aadd = Cadd) enabled
M1	1	YES	fix and float range test enabled
M2	0	NO	Reserved: must be cleared to 0
M3	0	NO	Input Bypass Mode disabled
M4	1	NO	Output Bypass Mode enabled
M5	1	YES	Overflow exception enabled
M6	1	NO	Coprocessor Load Mode enabled
M7	1	NO	Reserved: must be set to 1
M8	0	NO	FPEX active low and "sticky"
M10	1	NO	Reserved: must be set to 1
M11	1	NO	Internal Bypass Mode (Aadd = Badd) enabled

Figure 7. Mode selection table

XL-8232 Architecture, continued

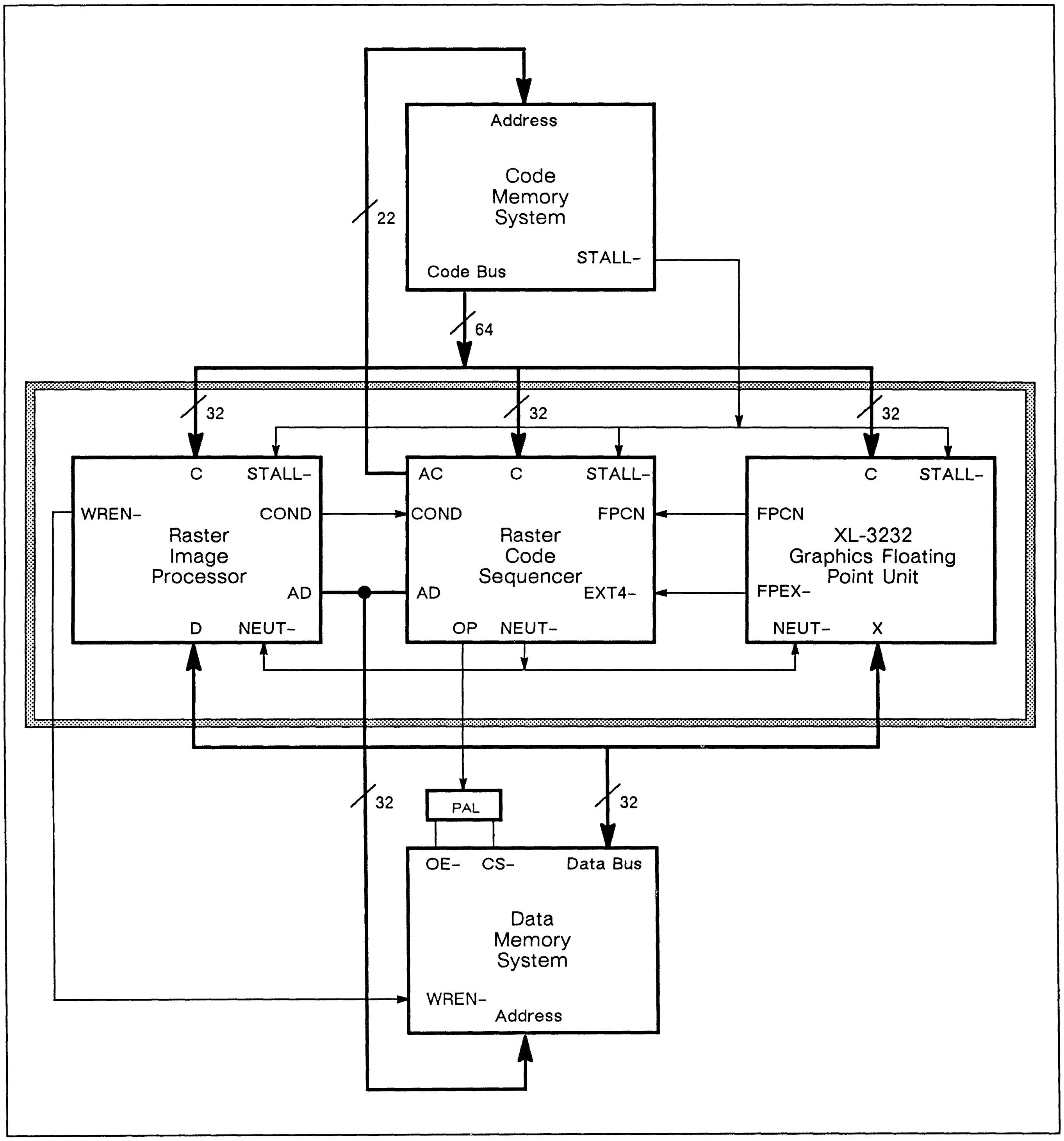


Figure 8. XL-8232 block diagram

XL-8232 Architecture, continued

CONDITIONS AND EXCEPTIONS

The XL-8232 provides several signals which transfer state information from the processing units (RIP, FPU) to the sequencer (RCS). These are either conditions, upon which the RCS may decide to branch; or exceptions, which require software intervention to recover gracefully.

The FPCN output on the XL-3232 must be connected to the FPCN input on the XL-8236. The FPCN signal is enabled by the $\mathrm{Encn}_{1..0}$ field in the instruction word to indicate whether the result of an operation is equal to zero, less than zero, or less than or equal to zero. The RCS may then execute a "branch on condition" instruction to selectively transfer program control according to the outcome of this comparison.

The FPEX– output on the XL-3232 must be connected the EXT4– interrupt input on the XL-8236 (even if you do not plan to use floating-point exceptions). If the overflow enable bit in the mode register is set, then any arithmetic operation that generates an invalid result can flag this exception to the RCS. The system software is expected to react appropriately to this interrupt.

NEUT– AND STALL–

The XL-8232 components all use the NEUT– and STALL–. These pins should be connected directly between the three chips in the XL-8232 processor (see figure 8).

NEUT– cancels the effect of the current instruction. The signal is generated by the RCS. It is normally used in the shadow of a delayed branch to prevent the instruction in the pipeline from having any effect on the state of the RIP and FPU.

STALL– cancels the effect of the next instruction. It should be generated by the code memory subsystem to indicate the delay or absence of the correct code word. This prevents any invalid operation that may be present on the code input at this time from affecting the state of the processor. It allows wait states to be inserted in code fetches, perhaps to allow for DRAM refresh or a code cache miss.

Register File

The XL-3232 has thirty-two 32-bit general-purpose registers. Each register can store either a single-precision IEEE value or a two's complement integer value.

PORTS

The register file has four ports, A, B, C and D. The A and B ports are read-only, the C port is write-only, and the D port is bi-directional. Each port can transfer a 32-bit data word on every clock cycle.

The A and B ports may be used to supply operands to the multiplier/accumulator and the divide look-up table. The C port receives the result of a previous operation. The D port communicates data between the register file and the external X port.

This organization allows I/O transfers to proceed in parallel with calculation, maximizing system performance.

REGISTER SELECTION

The registers that are to take part in each transfer are selected by the instruction word. The instruction format allows a register address to be supplied for each port. They are provided in the Aadd, Badd, Cadd and Dadd fields of the instruction. These fields are five bits in length, allowing each address to specify any of the thirty-two registers.

An instruction supplies the Aadd, Badd, and Dadd addresses to the register file during its first cycle and the Cadd address during its fourth cycle. This way a single instruction specifies all of the stages of an operation from initial source to ultimate destination.

It is possible for a register to be selected by more than one field in the same cycle, in which case the following rules apply:

1. If only read operations are to be performed on the register in question, then its value is copied to all of the necessary ports.
2. If two ports (C and D) attempt to write into the same register on the same cycle, the contents of the register will be left in an undefined state. Such contention must be avoided.
3. If a register is to be both read and written on the same cycle, the new value will be read after it is written to the register file.

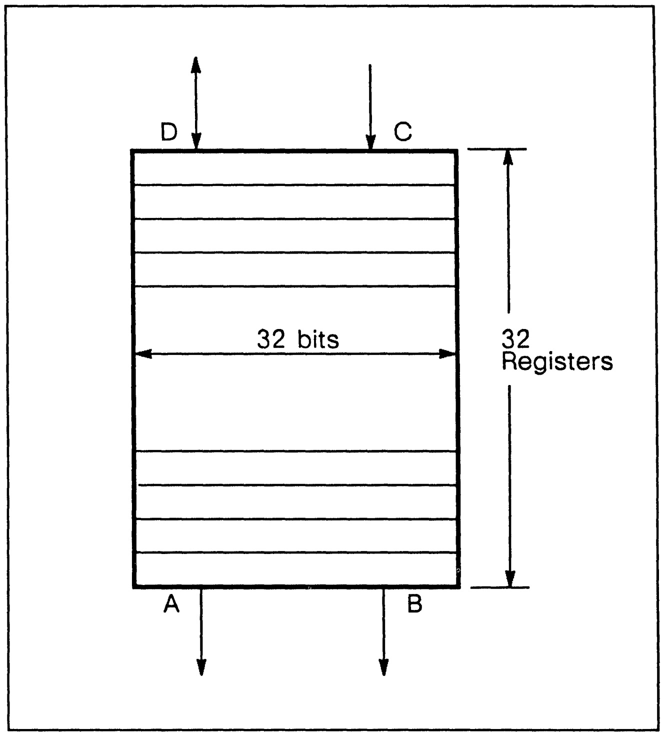


Figure 9. The four-port register file

READ/WRITE CONTROL

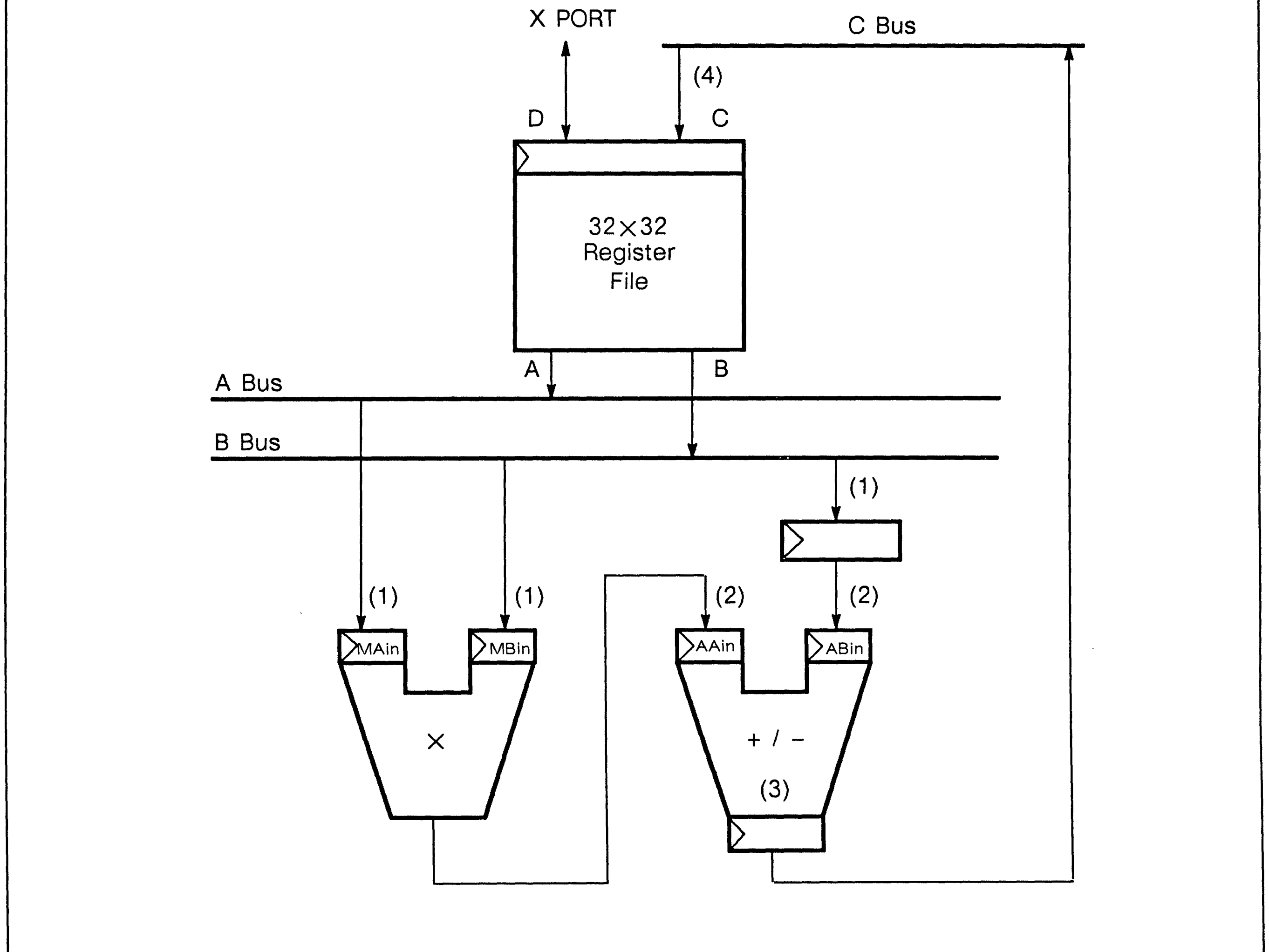
Two other fields in the instruction word affect the operation of the register file.

1. The Cwen- bit controls writing of results into the C port. When it is active (low), the result data is written on the fourth cycle of the operation. When writes are disabled, the contents of the register specified by Cadd remain unchanged.

   Register writes may be disabled either to direct a result to a Temporary Register or to allow arithmetic comparisons to modify the Status and Condition Registers without overwriting the contents of a general-purpose register.
2. The $IOCt_{1..0}$ bits control the direction of D port transfers (see page 24 for details). If the C port and the D port attempt to write to the same register file location on the same cycle the register contents are left undefined.

Multiplier/Accumulator

The XL-3232 has a pipelined multiplier/accumulator. These consist of a floating-point multiplier whose output is fed into a floating-point ALU (Arithmetic and Logic Unit). All multiplier/accumulator input and output ports can transfer 32-bit data values. Figures 10 and 11 show how operations are pipelined through the multiplier/accumulator.



Note: For clarity, many key features have been omitted from this diagram (see page 5 for more detail).

Figure 10. Simple example of multiplier/accumulator timing.

Multiplier/Accumulator, continued

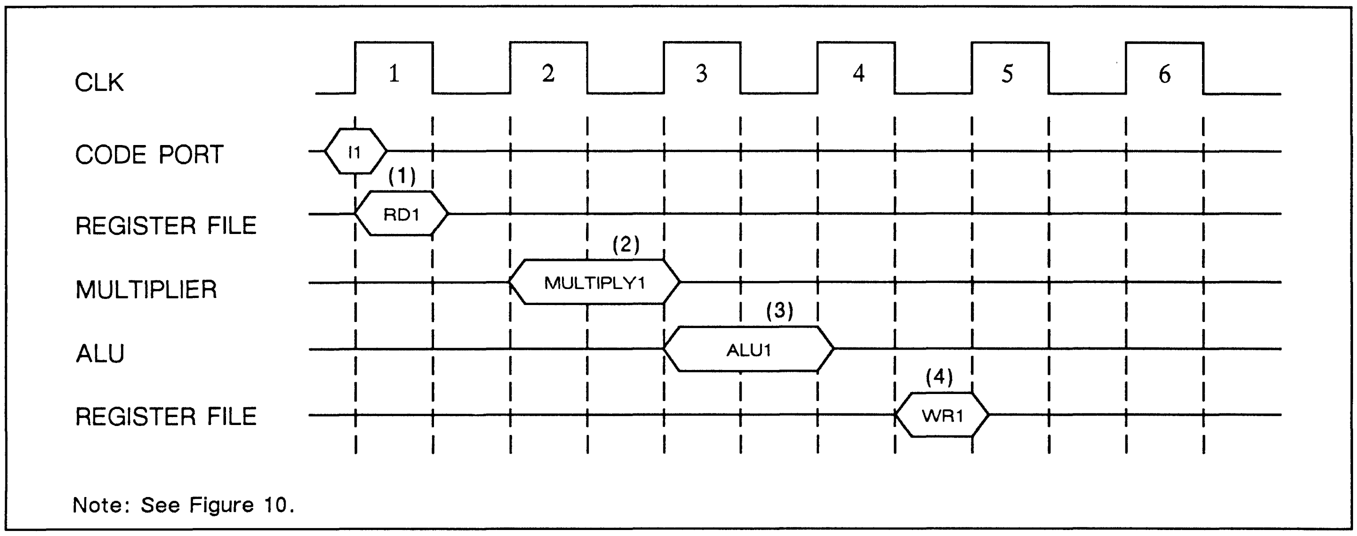


Figure 11. Timing of MAC operations

MULTIPLIER

The multiplier has two input ports, MAin and MBin. It has one output which can only be connected to the AAin port of the ALU. In the first cycle of an operation, operands are transferred from the register file and fed into MAin and MBin. The multiplication is completed on the second cycle. The intermediate result may be negated before it is passed to the ALU.

ALU

The ALU has two input ports, AAin and ABin. It has one output which is normally connected to the C bus. AAin may be connected to the multiplier's output, so that its result is fed into the ALU. Another operand is fed into the ABin port simultaneously. The ALU completes the function specified by an instruction during its third cycle. The final result is rounded and output to the C bus to be returned to the register file on the fourth cycle.

LATENCY

Because the multiplier/accumulator is pipelined, an operation can be initiated every cycle. The result of an operation is generated three cycles after it is initiated. On the fourth cycle, the result can be returned to the register file or fed straight back into the multiplier/accumulator.

Multiplier/Accumulator, continued

FUNCTION SELECTION

The multiplier/accumulator function is specified by the 3-bit field $F_{2..0}$ in the instruction word as outlined in the function select table (figure 12). A single instruction specifies all of the actions associated with one operation as it passes through the multiplier/accumulator.

When the $F_{2..0}$ field is (0, 0, 0) the operation to be performed is specified by the Badd field according to figure 13.

F2 F1 F0	MNEMONIC	OPERATION	DESCRIPTION
0 0 0	-	Miscellaneous	See figure 13
0 0 1	fsubr	Negate and add	-AAin + ABin
0 1 0	fsub	Subtract	AAin - ABin
0 1 1	fadd	Add	AAin + ABin
1 0 0	-	Reserved	
1 0 1	fmna	Multiply, negate and add	-(MAin × MBin) + ABin
1 1 0	fmns	Multiply, negate and subtract	-(MAin × MBin) - ABin
1 1 1	fmac	Multiply and accumulate	(MAin × MBin) + ABin

Figure 12. Function select field encoding

Badd4-0	MNEMONIC	OPERATION	DESCRIPTION
00000	fclsr	Clear Status Register	
00001	fstsr*	Read Status Register	
00010	-	Reserved	
00011	fmode	Load Mode Register	
00100	fabs	Absolute Value	|AAin|
00101	float	Fixed-to-Float	integer → IEEE
00110	fix	Float-to-Fixed	IEEE → integer
00111	flut	Look-up Operation	
01000-11111	-	Reserved	

* fstsr instructions must have their $IOCt_{1..0}$ field set to 10 to select a store.

Figure 13. Miscellaneous function select encoding.

Multiplier/Accumulator, continued

MULTIPLY/ACCUMULATE FUNCTIONS

The XL-3232 provides three multiply and accumulate functions. fmac multiplies MAin and MBin and then adds ABin. fmns multiplies MAin and MBin, negates the result and then subtracts ABin. fmna multiplies MAin and MBin, negates the result and then adds ABin.

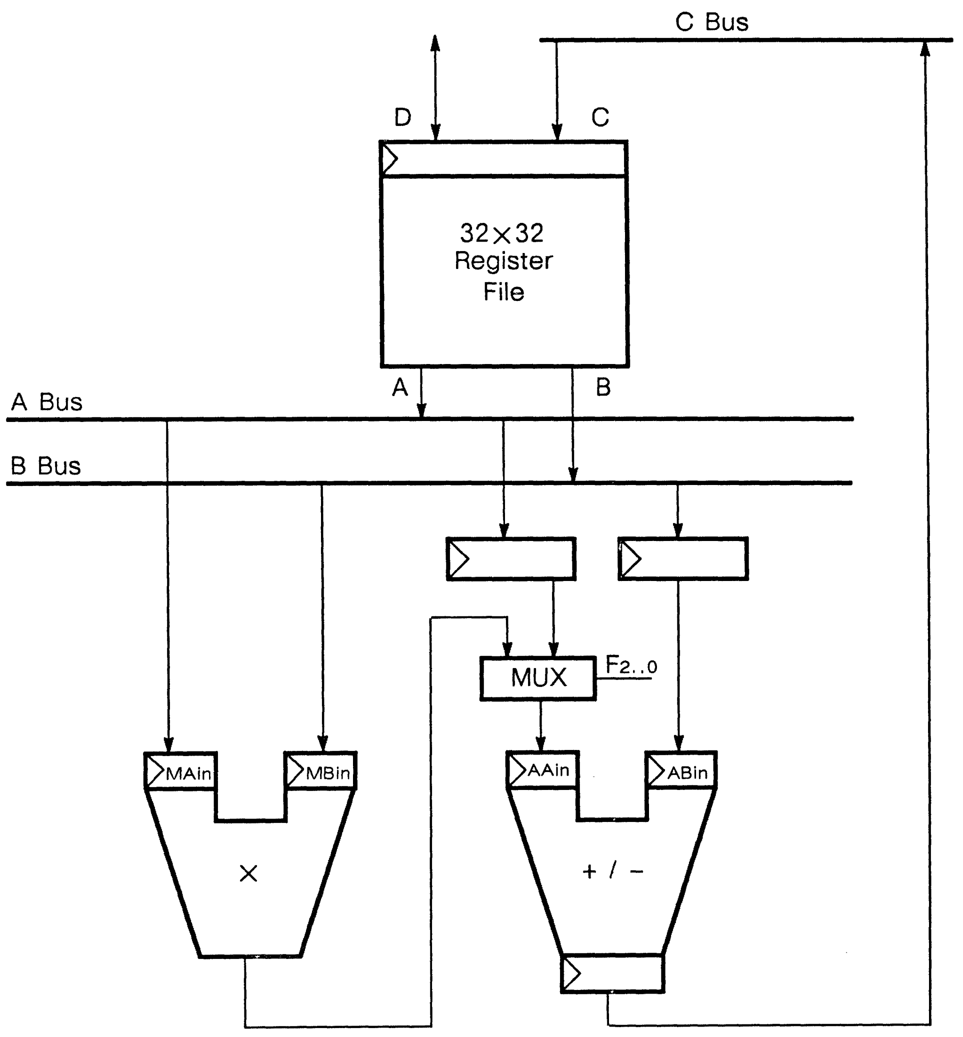
These functions are triadic (they have three input operands). If an IEEE multiply operation is required, the constant 0.0 should be selected as the ABin input.

ALU FUNCTIONS

The XL-3232 provides three dyadic "ALU only" functions. fadd adds AAin to ABin. fsub subtracts ABin from AAin. fsubr subtracts AAin from ABin.

The $F_{2..0}$ field determines whether the multiplier is bypassed and the ALU's input staged directly into the ALU (see figure 14).

These functions operate in the same number of cycles as the multiply and accumulate functions. This simplifies the programmer's model; every operation has the same latency.



Note: For clarity, many key features have been omitted from this diagram (see page 5 for more detail).

Figure 14. "ALU only" operations

MISCELLANEOUS FUNCTIONS

If the function field is equal to zero, then a miscellaneous ALU function will be selected according to the contents of the instruction's Badd field (see figure 13).

1. Flut is monadic (that is, it has a single input operand). It takes the value on the A bus as its operand and it returns an approximation to the inverse of this value onto the C bus on its fourth cycle. flut does not attempt to modify the Status, Condition, or Zero Registers. The Abin$_{2..0}$ field must be set to select the constant 0.0. (See page 37.)
2. Fix is a monadic "ALU only" function. It takes a single-precision IEEE format floating-point value on the A bus as its operand and returns a 24-bit, sign extended, two's complement integer onto the C bus on its fourth cycle. fix does not attempt to modify the Status or Zero Registers. It is the only instruction that produces an integer result. The Abin$_{2..0}$ field must be set to select the constant 0.0. (See page 39.)
3. Float is a monadic "ALU only" function. It takes a 24-bit, sign extended, two's complement value on the A bus as its operand and returns a single-precision IEEE format floating-point number onto the C bus on its fourth cycle. float does not attempt to modify the Status or Zero Registers. It is the only instruction that requires an integer operand. The Abin$_{2..0}$ field must be set to select the constant 0.0. (See page 39.)
4. Fabs is a monadic "ALU only" function. It takes the value on the A bus as its operand and returns its absolute value onto the C bus on its fourth cycle. fabs does not attempt to modify the Status, Condition, or Zero Registers. The Abin$_{2..0}$ field must be set to select the constant 0.0. As with the dyadic "ALU only" functions, it will replace denormalized operands with zero and NaNs with infinity.
5. Fmode loads the desired operating modes into the Mode Register (see page 34). Because this operation changes the timing of many operations, the results of the next three operations should be discarded.

   Fmode is not canceled by the NEUT– signal. It should not be executed in a branch shadow.
6. Fstsr copies the contents of the Status Register to the X port. It has the same timing as the other fstore operations. (See page 26.) The Cwen– bit must be set to prevent register writes, the Encn$_{1..0}$ field to disable updates of the FPCN pin, the IOCt$_{1..0}$ field to an fstore and that the result sent to the register file on its fourth cycle be discarded. fstsr ignores the register address fields.
7. Fclsr clears the contents of the Status Register to zero. (See page 26.) The Cwen– bit must be set to prevent register writes, the Encn$_{1..0}$ field to disable updates of the FPCN pin, the IOCt$_{1..0}$ to an I/O nop and that the result sent to the register file on its fourth cycle be discarded. fclsr ignores the register address fields.

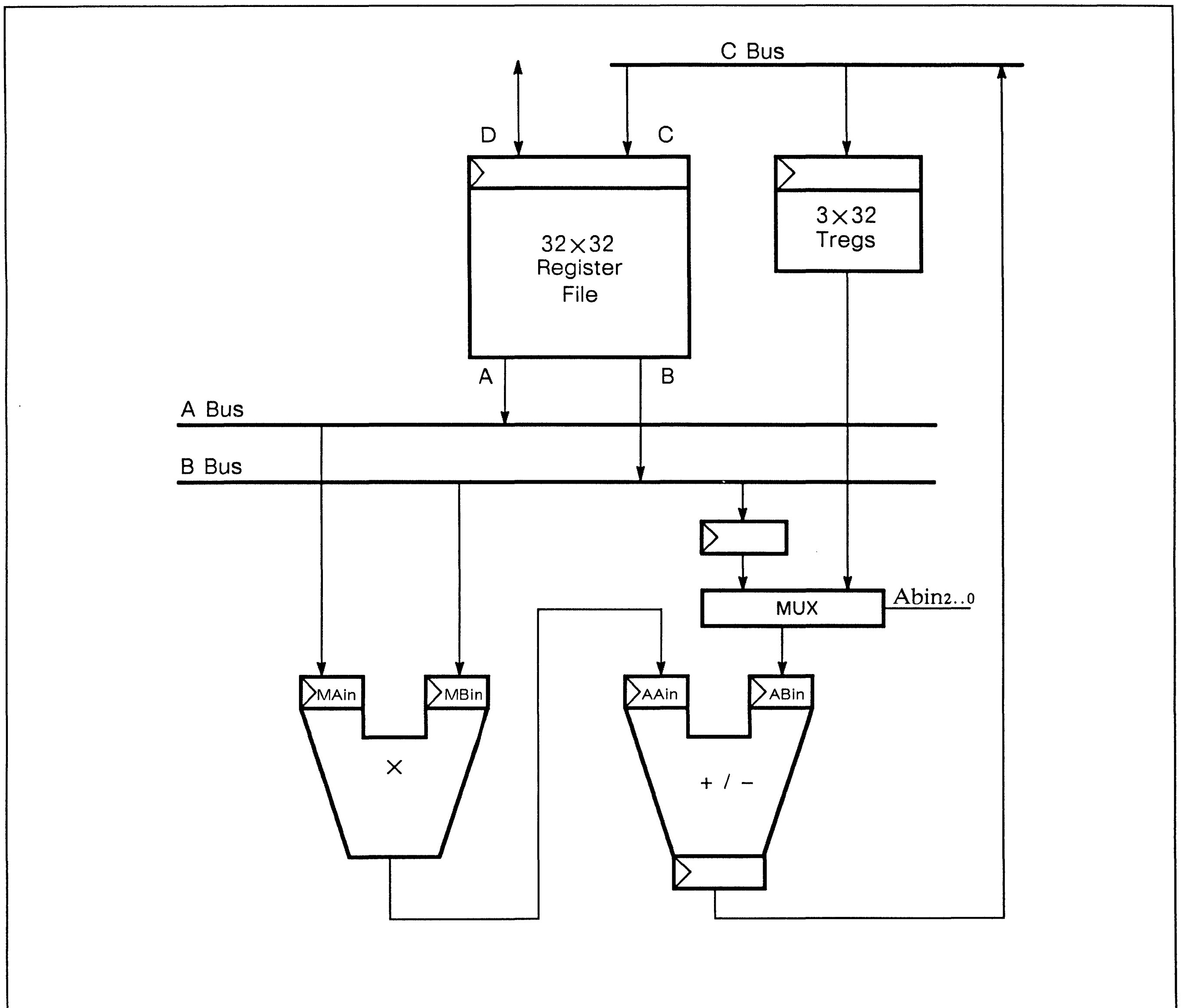
MULTIPLIER/ACCUMULATOR NOP

The XL-3232 does not have a dedicated nop instruction. Use fsub .f0, .f0, .f0 with the Cwen– bit set to disable register writes and the Encn$_{1..0}$ field cleared to disable FPCN updates. This choice of nop causes no state changes.

Temporary Registers

The XL-3232 includes three 32-bit temporary registers (Tregs). They allow values to be recirculated to the ALU without passing through the general-purpose register file. The Tregs are often used as accumulators during successive multiply/accumulate operations. They make it possible to perform a calculation of the form "$x = x \pm (y \times z)$" every cycle.

Figures 15 and 16 show how the Tregs are used to feedback operands to the multiplier/accumulator: figure 18 gives an example of a code sequence that does this.



Note: For clarity, many key features have been omitted from this diagram (see page 5 for more detail).

Figure 15. Use of temporary registers

Temporary Registers, continued

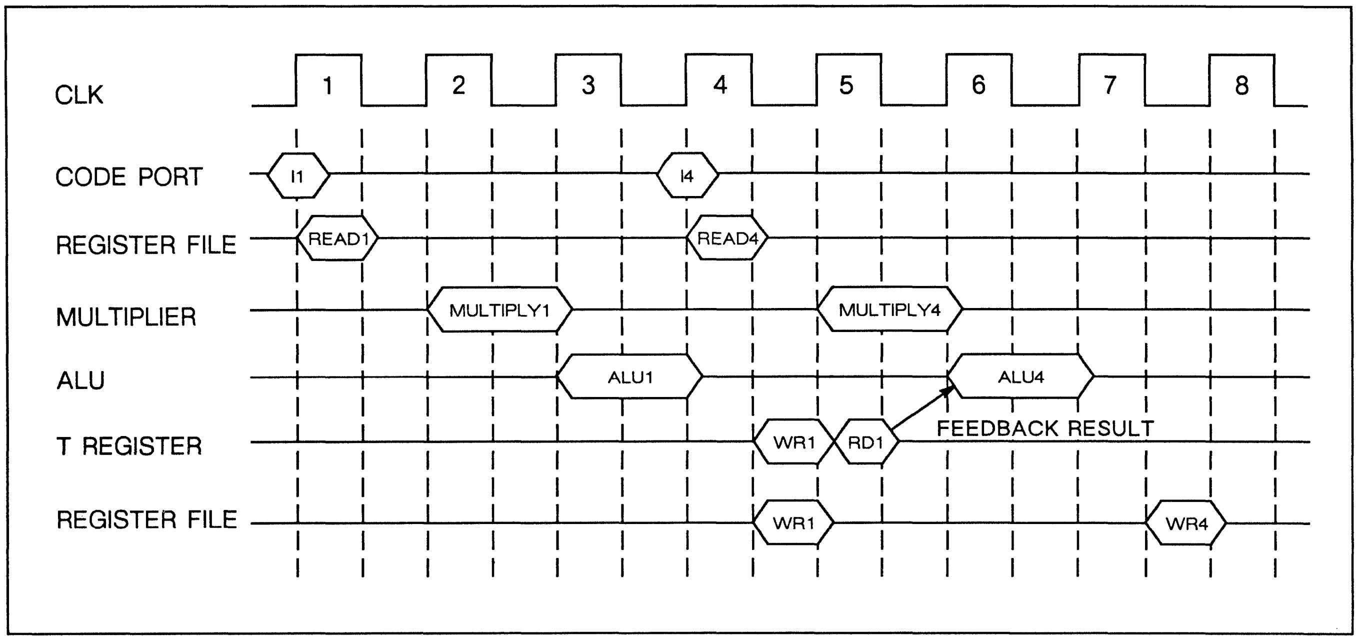


Figure 16. Temporary register timing

WRITING TO TEMPORARY REGISTERS

The instruction word contains a two-bit field, Adst1..0, that determines the destination of the ALU output (see figure 17).

The output of the ALU is always sent to the C bus. If no Treg is selected by the Adst1..0 field, then the result is returned only to the register selected by this instruction's Cadd field on its fourth cycle.

Adst1-0	RESULT DESTINATION
00	Treg3, C bus
01	Treg2, C bus
10	Treg1, C bus
11	C bus

Figure 17. ALU destination select field encoding

If the Adst1..0 field selects a Treg in addition to the C bus, it is loaded with the result on the fourth cycle of an operation, just as the Cadd register write occurs. On the next cycle, the contents of the Treg may be input directly to the ABin port. This is illustrated by the example shown in figure 18.

The Cwen- bit of the instruction that writes to the Treg may be set to 1 to prevent the write to the Cadd register. This increases the number of available general-purpose registers.

Temporary Registers, continued

READING TEMPORARY REGISTERS

The instruction word contains a three-bit field, Abin$_{2..0}$, that determines the source of the MAC's ABin input.

Three encodings select one of the Tregs (see figure 20). If a Treg is selected, it is copied to the ABin port on the second cycle of the operation.

The Treg may be be read on the cycle after it was written. In figure 18, for example, .t1 gets read during the second cycle of op #4. This is the fifth cycle of op #1.

```
.
.
.
op #1   fmac  .f0,  .f1,  .f0,  .t1
op #2   fmac  .f3,  .f4,  .t1,  .f5   —old .t1
op #3   fmac  .f6,  .f7,  .t1,  .f8   —Illegal
op #4   fmac  .f9,  .f10, .t1,  .f11  —new .t1
.
.
.
```

Notes:

A Treg cannot be both written and read in the same cycle. Op #3 *must never* attempt to read the Treg written by op #1.

To make this code interruptable, op #2 and op #3 should not specify .t1 as an operand.

Full details of this syntax are given on page 32.

Figure 18. Use of temporary registers

Internal Data Routing

INTERNAL BUSES

The three main internal buses, A, B and C, move data between the major functional units of the XL-3232. Each of these buses is 32 bits wide and can carry one word per cycle.

The A bus usually carries operands from the register file to the multiplier/accumulator or the divide look-up table.

The B bus usually carries operands from the register file to the multiplier/accumulator.

The C bus usually carries multiplier/accumulator or flut results back to the general-purpose register file. It may also be used to feed the results directly to the X port.

Mbin-	INPUT
0	B bus
1	C bus

Figure 19. Multiplier input port select field encoding

Abin2-0	INPUT
000	C bus
001	B bus
010	Treg2
011	Treg1
100	Treg3
101	Reserved
110	2.0
111	0.0

Figure 20. ALU input select field encoding

MULTIPLIER/ACCUMULATOR INPUT PORTS

The multiplier/accumulator has four input ports, MAin, MBin, AAin and ABin. These ports can each receive a 32-bit word per cycle. They all have multiplexers which may be connected to various input sources. The possible selections for each port are described below:

1. MAin usually obtains input from the A bus. Results may be fed from the C bus to the A bus and then into MAin.
2. MBin usually obtains its input from the B bus. Results may be fed from the C bus to the B bus and then into MBin.

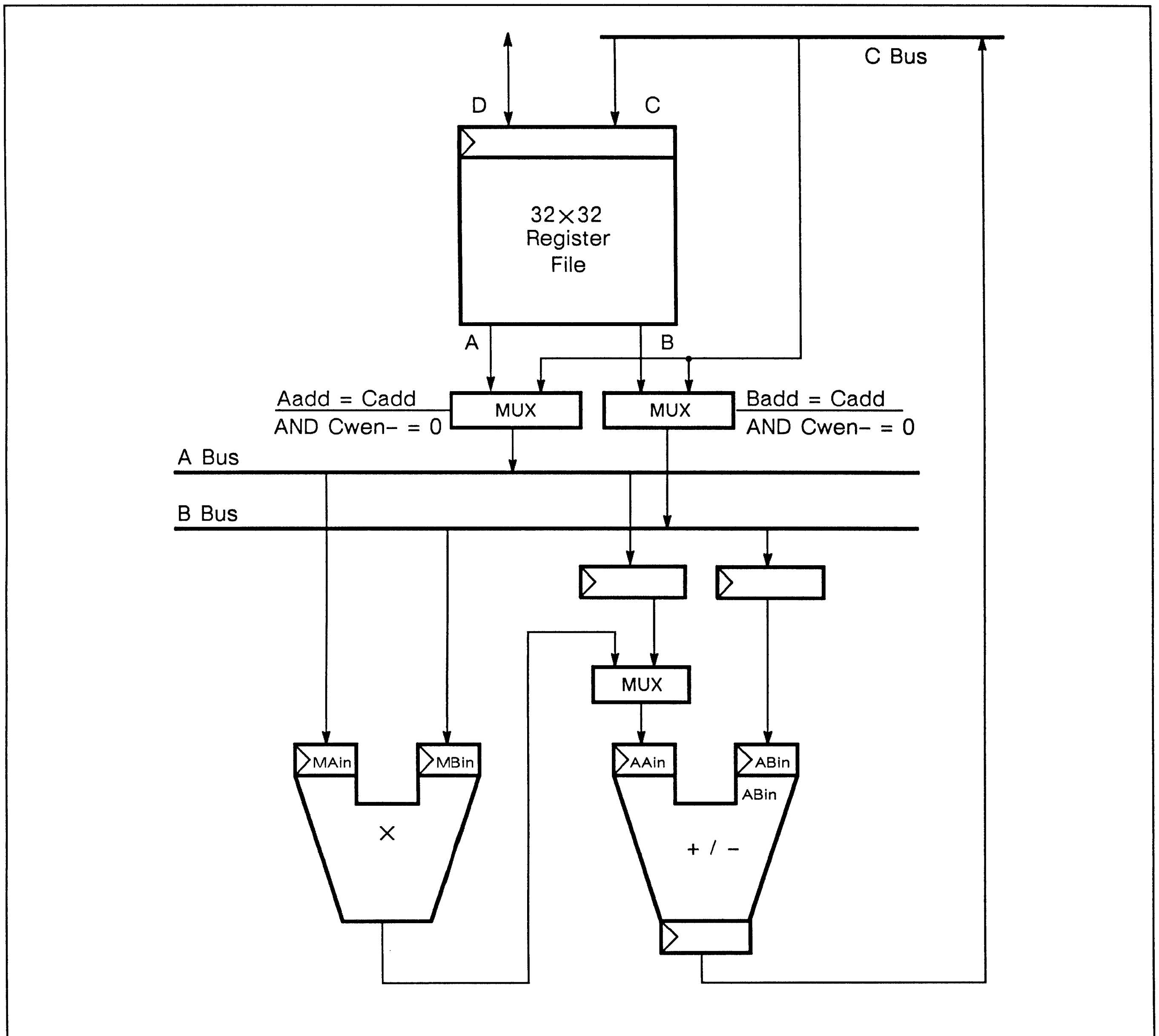
   Alternatively, the C bus can be reversed so that it is carrying inputs from the X port to the MBin port. This prevents the C bus from being used to return results to the register file and is done in conjunction with the floadrc operation. MBin must select the C bus (see figure 19).
3. AAin usually obtains its input from the A bus. Results may be fed from the C bus to the A bus and then into AAin.
4. ABin usually obtains its input from the B bus. Results may be fed from the C bus to the B bus and then into ABin. Y port inputs may be enabled onto the B bus and then into the ABin.

   The three-bit $Abin_{2..0}$ field in the instruction word selects between input from the B bus (as above), input of the constants 0.0 or 2.0, input from one of the Tregs, or input from the C bus (as below) according to figure 20.

   Alternatively, the C bus can be reversed so that it is carrying inputs from the X port to the MBin port. This prevents the C bus from being used to return results to the register file and is done in conjunction with the floadrc operation. ABin must select the C bus. When this pathway is used, the ABin data is not delayed.

Internal Data Routing, continued

If the function code specifies an operation that uses the multiplier, it directs data to the MAin or MBin ports. If the multiplier is not used, (that is, in "ALU only" operations), then the data is sent to the AAin or ABin ports automatically. The XL-3232 is designed to maintain a consistent latency regardless of the type of operation.



Note: For clarity, many key features have been omitted from this diagram (see page 5 for more detail).

Figure 21. Internal bypass routes

Internal Data Routing, continued

INTERNAL BYPASS MODE

A code sequence often specifies the result of one operation to be the operand of a subsequent operation. The XL-3232 contains internal data paths called *bypass paths* that allow the result of one operation to be used as the input to another without the delay of writing it to the register file and reading it out again. This operation is invisible to the programmer; it simply guarantees that three cycles after the start of an operation, the result is available wherever you need it. This is done through bypass muxes.

These multiplexers operate by comparing the Aadd and Badd address fields to the Cadd address field as they are presented to the register file on each cycle. If Aadd = Cadd, then the value on the C bus is copied to the A bus; and if Badd = Cadd, then the value on the C bus is copied to the B bus. Enough time remains for that value to be latched into a multiplier/accumulator input port before the end of the cycle. In the example, the Cadd field of op #1 matches the Aadd field of op #4 as they are compared on the fourth cycle of op #1, the bypass from C to A buses is opened and op #4 can proceed immediately with the new data. During the same cycle, the result is copied into the Cadd register as usual so that the register file remains consistent with the data values in use.

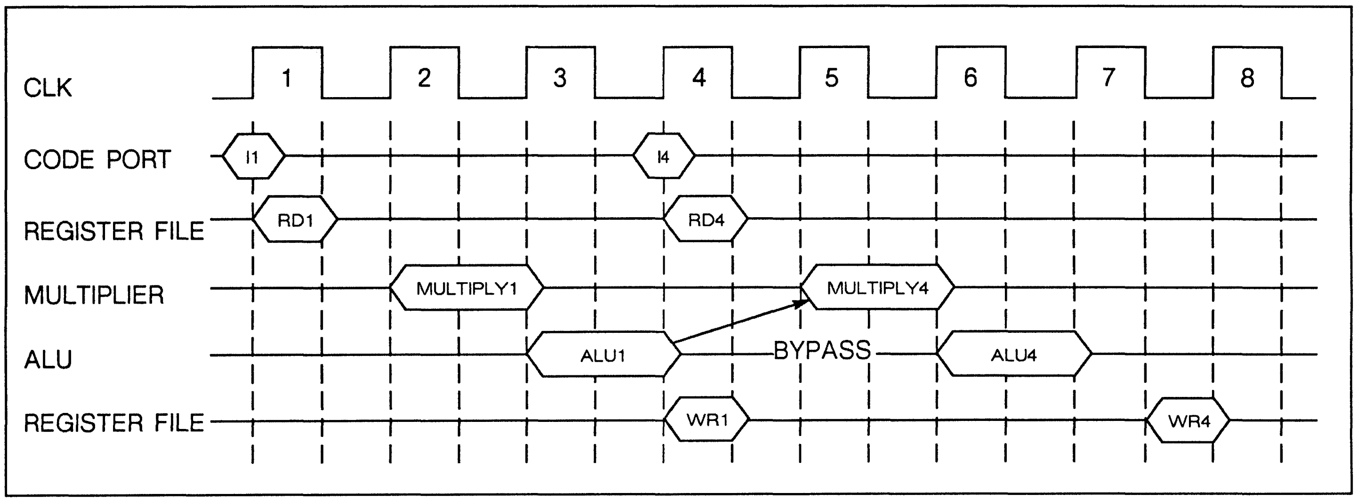


Figure 22. The timing of Internal Bypass Mode

Internal Data Routing, continued

TEMPORARY REGISTERS

To complete the range of alternative routes available for feeding results back to the multiplier/accumulator, the Treg code example first given in figure 18 is repeated here.

The temporary register option has the same timing as other data paths, but only feeds back to the ABin port. The Tregs bring an extra source of operands to the multiplier/accumulator, allowing operations of the form "$x = x \pm (y \times z)$" to be executed in a single cycle.

Many of the code examples given read a register after the instruction that modifies it has been initiated. While such code sequences are valid, they are uninterruptable. More detailed coverage of interruptable code may be found on page 32.

The data I/O port (X port) is 32 bits wide. It can all transfer one word per cycle. The memory-to-memory latency can be as low as five cycles (two more than the register-to-register latency). The output bus may be disabled by de-asserting its asynchronous output enable signals.

```
.
.
.
op #1   fmac  .f0,  .f1,  .f0,  .t1
op #2   fmac  .f3,  .f4,  .t2,  .f5
op #3   fmac  .f6,  .f7,  .t3,  .f8
op #4   fmac  .f9,  .f10, .t1,  .f11  —new .t1
.
.
.
```

Notes:

A Treg cannot be both written and read in the same cycle. Op #3 *must never* attempt to read the Treg written by op #1.

Full details of this syntax are given on page 32.

Figure 23. Use of temporary registers

Input/output

THE X PORT: NORMAL USAGE

The X port normally transfers data to and from the register file D port. The Dadd field selects the register in question and the IOCt1..0 field in the instruction word controls the transaction. These I/O transfers always begin during the first cycle of an operation. See figure 24 for the IOCt1..0 encoding scheme and figure 25 for normal I/O timing.

The fload operation loads the value at the X port pins into the register selected by Dadd. It is completed by the end of the first cycle. If the register is read on the same cycle, its previous contents will be output.

The fstore operation stores the contents of the register selected by Dadd to the X port output register (XDoutR) during the first cycle. This value is driven onto the X port pins on the second cycle. The fstore operation drives the X pads during most of its second cycle and at the start of its third cycle. Input data may not be applied to the pins until partway through its third cycle. The OEX- pin can asynchronously disable the output at any time.

The floadrc operation is described on page NO TAG.

The I/O nop operation simply disables the X port and ignores any input. It does not prevent the multiplier/accumulator from writing to registers or modifying the state of the condition and exception outputs.

Note: An fload should not follow a fstore immediately. At least one I/O nop cycle must be inserted between them.

IOCt1-0	OPERATION
00	I/O nop
01	floadrc
10	fstore
11	fload

Figure 24. X port I/O control field encoding

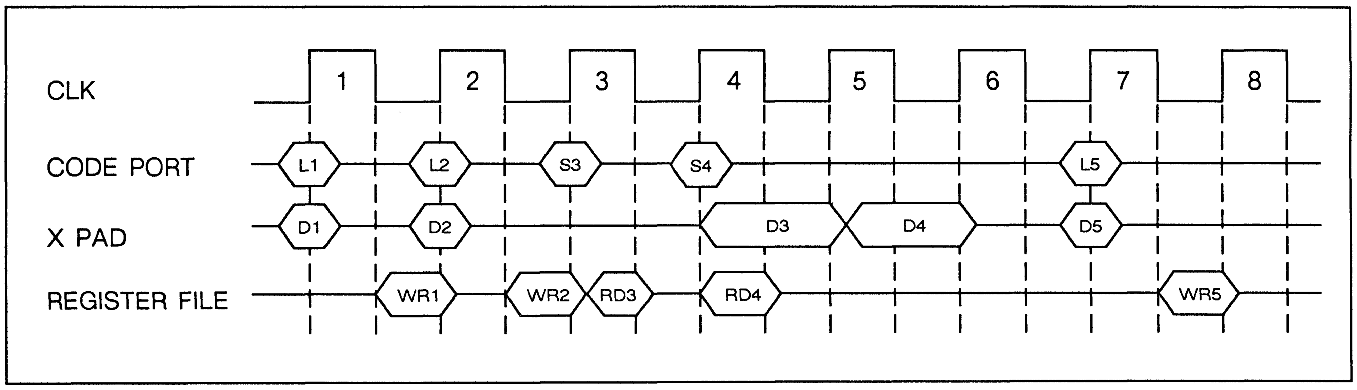


Figure 25. Normal X port I/O timing

OUTPUT BYPASSING

During a normal fstore operation the multiplier/accumulator result must be written to a register one cycle before it can be output to the X port. If, however, Cadd = Dadd; the multiplier/accumulator result is sent to both the register selected by Cadd and the X port output register (XCoutR) on the same cycle.

The fstore instruction that specifies the Dadd must start execution on the fourth cycle of the arithmetic instruction that specified the Cadd. The output appears at the X port pins during the second cycle of the fstore instruction.

System Interfacing

Certain signals on the XL-3232 are provided to communicate control information to and from the other parts of a system.

Two outputs, FPCN and ZERO, indicate the condition of an operation. They can be sent to a sequencer to control instruction branching.

One output, FPEX, signals the occurrence of arithmetic overflow. It can be used to interrupt a host processor to request corrective action.

Two inputs, NEUT– and STALL, allow the effects of instructions fed into the C port to be canceled. They can be used to make the XL-3232 respond correctly to page faults, interrupts or other system requests.

CONDITION AND ZERO

The XL-3232 has a Condition Register and a Zero Register. The multiplier/accumulator attempts to modify the contents of these registers on every cycle.

The instruction word includes a two-bit condition select field, $Encn_{1..0}$, which selectively allows the multiplier/accumulator to succeed in updating the contents of these registers. If both bits are cleared, then the previous state of the registers remains unchanged.

Most functions update the Condition Register according to the sign and magnitude of their result. Miscellaneous functions may set the register for other reasons (see figure 26).

$Encn_{1..0}$ determines the exact condition that will set the Condition Register for each instruction. This allows any of the common comparisons to be made in one operation. Figure 27 gives the bit encoding.

If the result of an operation is exactly equal to zero and the $Encn_{1..0}$ field is not (0,0); the Zero Register is set to 1. If the result is not zero and $Encn_{1..0}$ is not (0,0); the Zero Register is cleared to 0. If $Encn_{1..0}$ is (0,0); the contents of the Zero Register remain the same.

The contents of the Zero and Condition Registers are copied to the ZERO and FPCN outputs respectively on the fourth cycle of the operation, just as the general-purpose register file write occurs. Bypassing does not affect the timing of these signals. These outputs always drive a logic 0 or 1.

FUNCTION	SET CONDITION REGISTER	SET ZERO REGISTER	SET STATUS REGISTER
fmac	$N \leq 0$, $N < 0$, $N = 0$	$N = 0$	exp(N) > = 255
fmns	$N \leq 0$, $N < 0$, $N = 0$	$N = 0$	exp(N) > = 255
fmna	$N \leq 0$, $N < 0$, $N = 0$	$N = 0$	exp(N) > = 255
fadd	$N \leq 0$, $N < 0$, $N = 0$	$N = 0$	exp(N) > = 255
fsub	$N \leq 0$, $N < 0$, $N = 0$	$N = 0$	exp(N) > = 255
fsubr	$N \leq 0$, $N < 0$, $N = 0$	$N = 0$	exp(N) > = 255
flut	–	–	–
fabs*	$N \leq 0$ (only true when N=0)	–	–
fix*	$\lvert M \rvert > 2^{22}$	–	–
float*	$M < (-2^{23})$ or $M > (2^{23} - 1)$	–	–
fnop	–	–	–

N is the result of an operation; M is the operand.
* Fix and float only test for range overflow when M1 = 1. Fabs only tests for zero when M1 = 0.

Figure 26. Effect of functions on Condition, Zero, and Status Register

Encn1	Encn0	SET ZERO REGISTER	SET CONDITION REGISTER
0	0	-	-
0	1	N=0	N $\leq$ 0
1	0	N=0	N<0
1	1	N=0	N=0

Note: Encn1 and Encn0 must be zero for fnop.
Condition Register is set by fix or float when $Encn_{1..0}$ is (0,1) and the operand exceeds the permitted range.

Figure 27. $Encn_{1..0}$ encoding

STATUS AND EXCEPTIONS

The XL-3232 has a Status Register. If an operation produces a result that is too large to be represented in the IEEE single-precision floating-point format the multiplier/accumulator attempts to set the Status Register to 1.

The Mode Register includes an exception control bit, M5 (see page 34). If M5 is set to 1 and an overflow occurs, the Status Register is set to 1. If M5 is cleared to 0, the Status Register is cleared to 0. If M5 is subsequently set to re-enable overflows, the Status Register will contain 0.

The contents of the Status Register are copied to the FPEX output on the fourth cycle of the operation, just as the register file write occurs.

The Status Register is "sticky"; once set it will remain so until an fclsr operation is performed.

Two miscellaneous functions, fstsr and fclsr, allow the Status Register to be read at the X port or for it to be cleared. They take effect during their first cycle. If the fstsr operation is performed, then the $IOCt_{1..0}$ field must specify a 'store' and its timing is the same as a normal store operation (see figure 28). If the fclsr operation is performed, it is complete by the end of its first cycle. Only the least-significant bit of the Status Register is guaranteed; the other 31 bits are undefined.

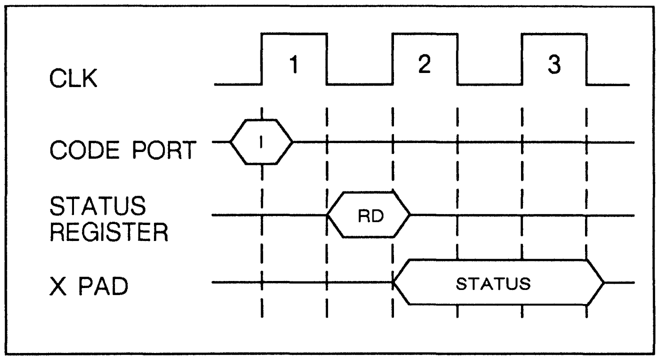


Figure 28. fstsr timing

System Interfacing, continued

LOADING DATA

The data applied to the X port is not sampled until late in the first cycle. Time still remains to write the Dadd register before the end of this cycle. As usual, the next instruction can use this data value as one of its operands.

NEUT- AND STALL-

These inputs allow a system to modify the effect of certain instructions dynamically.

NEUT-

Neutralize is used to prevent the execution of instructions in the shadow of a delayed branch operation or during an interrupt service cycle.

In a system where the sequencer supports delayed branching, it will present the next instruction to the C port as it decides whether to take a branch. If the branch is taken, this instruction must be cancelled before it has any effect on the state of the system. Similarly, if an interrupt occurs, the instruction due to be executed can be cancelled in order to branch to an interrupt service routine. The cancelled instruction is resubmitted for execution on return from interrupt.

The neutralize signal cancels the effect of the current instruction. It prevents the result of this instruction from being written into the register file or temporary registers. It has no effect on fload or fstore operations. This signal is sampled on the rising edge of the cycle after the current instruction was fed into the C port.

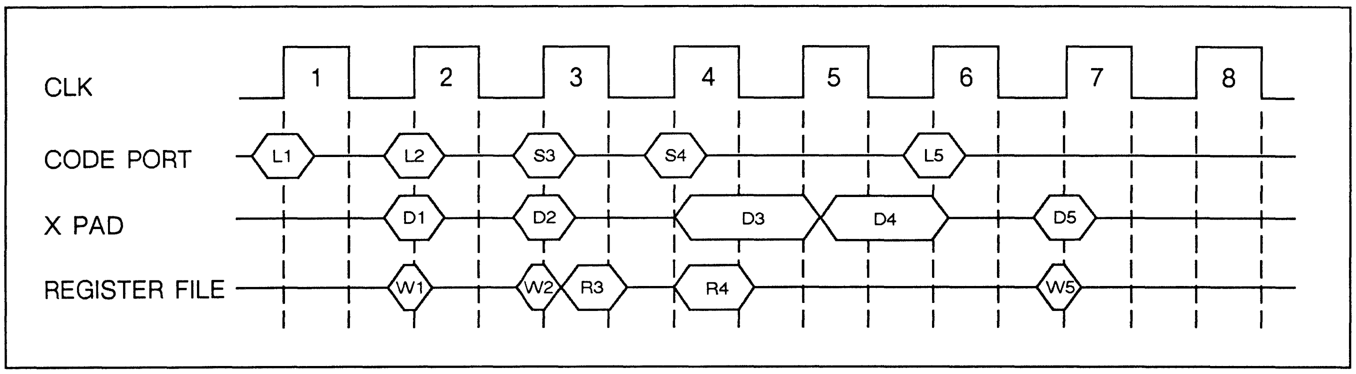


Figure 29. Coprocessor Load Mode timing

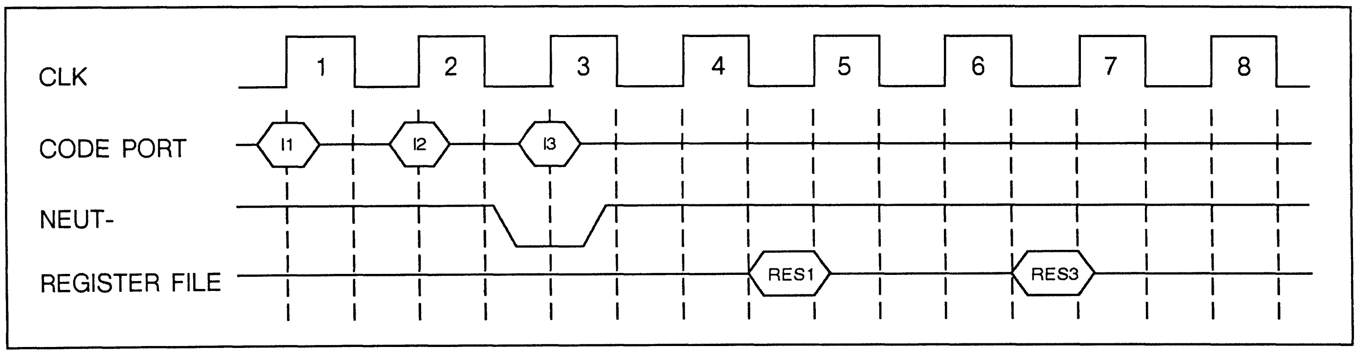


Figure 30. NEUT- timing

STALL–

STALL– is used to hold off execution until a valid code word is present when the code word is delayed (as in a code memory refresh cycle) or absent (as in a page fault). The next operation can be continually stalled until the correct instruction word is presented to the C port.

The STALL– signal cancels the effect of the next instruction. It prevents the result of this instruction from being written into the register file or temporary registers. It also cancels fload and fstore operations. This signal is sampled at the same time as the next instruction is fed into the C port.

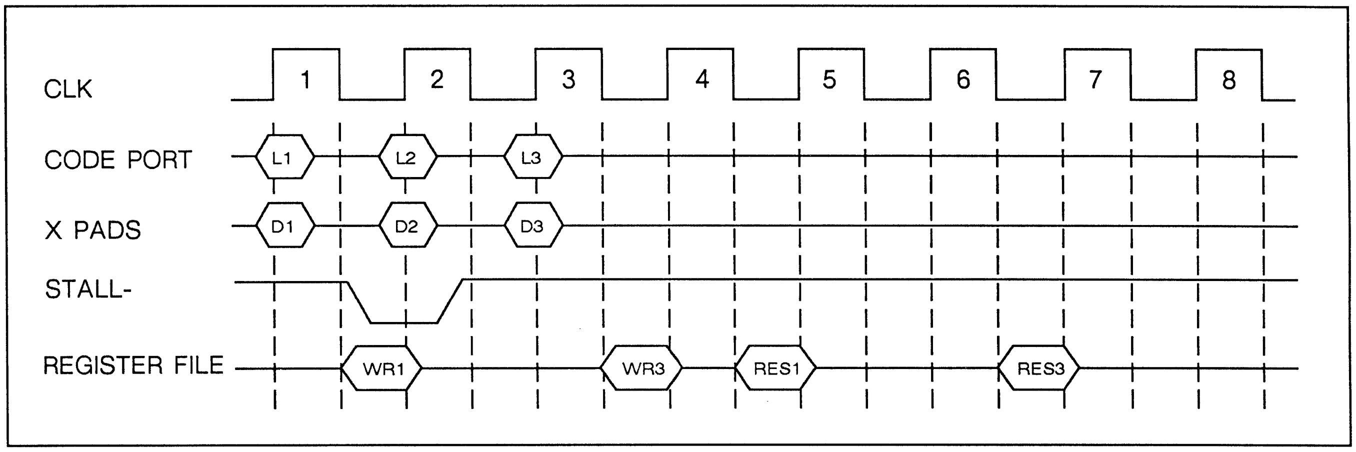


Figure 31. STALL– timing (including fload)

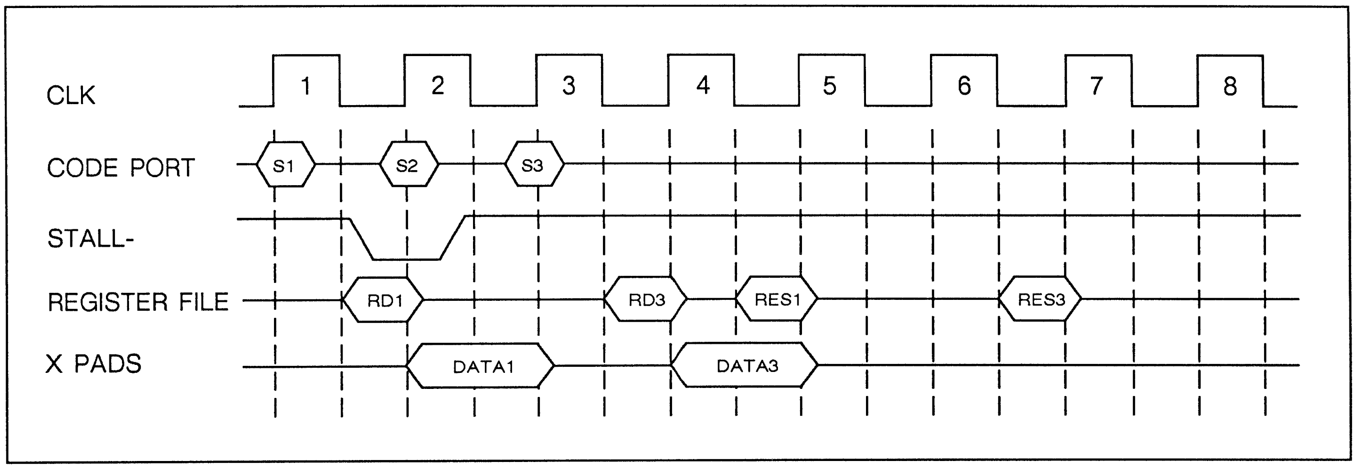


Figure 32. STALL– timing (including fstore)

Instruction Set

C BIT	FIELD	OPERATION
0	Encn0	Condition Output Select
1	Encn1	
2	Mbin-	MBin Input Select
3	Adst0	ALU Destination Select
4	Adst1	
5	Abin0	ABin Input Select
6	Abin1	
7	Abin2	
8	Dadd0	D Port Register Address
9	Dadd1	
10	Dadd2	
11	Dadd3	
12	Dadd4	
13	IOCt0	I/O Control
14	IOCt1	
15	Cwen-	C Port Write Enable
16	Cadd0	C Port Register Address
17	Cadd1	
18	Cadd2	
19	Cadd3	
20	Cadd4	
21	Badd0	B Port Register Address
22	Badd1	
23	Badd2	
24	Badd3	
25	Badd4	
26	Aadd0	A Port Register Address
27	Aadd1	
28	Aadd2	
29	Aadd3	
30	Aadd4	
31	F0	Function Code
32	F1	
33	F2	

Figure 33. Instruction format

FORMAT

The XL-3232 has a 34-bit instruction word. Refer to figure 33 for the location of each field in the instruction word.

F FIELD

Function control ($F_{2..0}$) field. Selects function to be performed on this instruction's operands.

AADD FIELD

A register address ($Aadd_{4..0}$) field. Selects location in general-purpose register file to be read out via the A port.

BADD FIELD

B register address ($Badd_{4..0}$) field. Selects location in general-purpose register file to be read out via the B port. Also encodes miscellaneous functions that require only one operand.

CADD FIELD

C register address ($Cadd_{4..0}$) field. Selects location in general-purpose register file to be written into via the C port.

CWEN- BIT

C port write enable bit. Active low.

IOCT FIELD

I/O control ($IOCt_{1..0}$) field. Selects type of I/O transfer performed via D port.

DADD FIELD

D register address ($Dadd_{4..0}$) field. Selects location in general-purpose register file to be read out or written into via the D port.

ABIN FIELD

ALU input multiplexer input control ($Abin_{2..0}$) field. Selects the input source for ALU.

ADST FIELD

ALU output destination control ($Adst_{1..0}$) field. Selects output destination for ALU. May be the C bus or the C bus and a Temporary Register.

MBIN- BIT

Must be tied to GND.

ENCN FIELD

Condition select ($Encn_{1..0}$) field. Enables a selectable combination of the condition and zero flags onto the FPCN output.

All of the actions specified by these fields are defined in the same instruction word. In this way, all of the stages of an operation, from supplying its operands to storing its results back into a register, are specified together.

Instruction Set, continued

MNEMONICS

The mnemonics shown in figures 34 through 36 are those used to control the XL-3232 in the XL-8200 programming environment. They are given here to simplify understanding of the programming model and to provide a syntax in which to present the programming examples.

I/O SELECTION	OPERAND	
	Source	Destination
fload	X port	.f0-31
fstore	.f0-31	X port
Note: If no I/O operation is specified, an I/O nop will be selected in the IOCt1..0 instruction field.		

Figure 34. Recommended mnemonics

OPERAND NOTATION	DESCRIPTION
.f0–31	Thirty-two, 32-bit general purpose registers
0	Constant "0.0"
2	Constant "2.0"
.t1-3	Three temporary registers

Figure 35. Recommended mnemonics

FUNCTION	OPERAND SELECTIONS			
	SOURCE (Aadd)	SOURCE (Badd)	SOURCE (Tregs)	DESTINATION (Cadd)
fmac	.f0-31	.f0-31	0, 2, or .t1–3	.f0-31 and/or .t1-3
fmns	.f0-31	.f0-31	0, 2, or .t1–3	.f0-31 and/or .t1-3
fmna	.f0-31	.f0-31	0, 2, or .t1–3	.f0-31 and/or .t1-3
fadd	.f0-31	.f0-31, 0, 2, or .t1-3		.f0-31 and/or .t1-3
fsub	.f0-31	.f0-31, 0, 2, or .t1-3		.f0-31 and/or .t1-3
fsubr	.f0-31	.f0-31, 0, 2, or .t1-3		.f0-31 and/or .t1-3
flut	.f0-31			.f0-31
fabs	.f0-31			.f0-31 and/or .t1-3
fix	.f0-31			.f0-31 and/or .t1-3
float	.f0-31			.f0-31 and/or .t1-3
fnop	–			

Figure 36. Recommended mnemonics

CODE CONSTRAINTS

The following set of rules prevents illegal code sequences:

1. All instructions must avoid writing to the Cadd register and Dadd register simultaneously. Thus no fload operation with Dadd = .fx may start on the fourth cycle of an operation with Cadd = .fx.

```
.
.
.
op #1 fadd  .f?, .f?,  .fx
op #2 fadd  .f?, .f?,  .f?;  fload.fx
op #3 fadd  .f?, .f?,  .f?;  fload.fx
op #4 fadd  .f?, .f?,  .f?;  fload.fx   —Illegal
op #5 fadd  .f?, .f?,  .f?;  fload.fx
.
.
.
```

Figure 37.

2. Because the X port output is driven on the cycle after an fstore operation is specified, an fload cannot follow an fstore immediately. At least one I/O nop must intervene.

```
.
.
.
op #1 fadd  .f?, .f?,  .f?;  .fstore.f?
op #2 fadd  .f?, .f?,  .f?;  .fload.f?  —Illegal
op #3 fadd  .f?, .f?,  .f?;  .fload.f?
.
.
.
```

Figure 38. Coprocessor Load Mode enabled, M6=1

3. No temporary register can be written and read on the same cycle. Thus no operation that selects .tx as an operand register may start on the third cycle of an operation with Cadd = .tx.

```
.
.
.
op #1 fmac  .f?,  .f?,  .f0,  .tx
op #2 fmac  .f?,  .f?,  .f0,  .f?
op #3 fmac  .f?,  .f?,  .tx,  .f?  —Illegal
.
.
.
```

Figure 39.

Instruction Set, continued

4. No operation with Aadd or Badd = .fx may start on or after the first cycle and before the fourth cycle of an operation with Cadd = .fx. Thus no instruction may use the same register for both source and destination.

5. No operation with Aadd or Badd = .tx may start on or after the first cycle and before the fourth cycle of an operation with Cadd = .tx. Thus no instruction may use the same Treg for both source and destination.

```
.
.
.
op #1 fadd  .f?, .f?, .fx
op #2 fadd  .fx, .f?, .f?  —Illegal
op #3 fadd  .fx, .f?, .f?  —Illegal
op #4 fadd  .fx, .f?, .f?  —Illegal
op #5 fadd  .fx, .f?, .f?
.
.
.
```

Figure 40.

```
.
.
.
op #1 fmac .f?, .f?, .f0, .tx
op #2 fmac .f?, .f?, .tx, .f?  —Illegal
op #3 fmac .f?, .f?, .tx, .f?  —Illegal
op #4 fmac .f?, .f?, .tx, .f?
.
.
.
```

Figure 41.

6. No operation with Aadd or Badd = .fx may start on the same cycle as an fload where Dadd = .fx.

```
.
.
.
op #1 fadd  .fx, .f?, .f?; fload.fx —Illegal
op #2 fadd  .f?, .f?, .f?
.
.
.
```

Figure 42.

7. The NEUT– line does not cancel floads and fstore, so when it is used to cancel the effect of an instruction in the shadow of a delayed branch operation, this instruction should not perform I/O transfers. (This is not necessary when NEUT– is asserted during an interrupt response cycle because the cancelled instruction is resubmitted for execution.)

In the examples, the notation .f? is used to indicate any register except .fx.

Initialization

MODE REGISTER

The Mode Register controls which of the special modes are enabled. Normally, it is initialized to the desired state and is not subsequently altered. Most bits are reserved and must be set to the value specified in figure 43.

MODE BIT	LOGIC VALUE	OPTIONAL?	DESCRIPTION
M0	1	NO	Internal Bypass Mode (Aadd = Cadd) enabled
M1	1	YES	fix and float range test enabled
M2	0	NO	Reserved: must be cleared to 0
M3	0	NO	Input Bypass Mode disabled
M4	1	NO	Output Bypass Mode enabled
M5	1	YES	Overflow exception enabled
M6	1	NO	Coprocessor Load Mode enabled
M7	1	NO	Reserved: must be set to 1
M8	0	NO	FPEX active low and "sticky"
M10	1	NO	Reserved: must be set to 1
M11	1	NO	Internal Bypass Mode (Aadd = Badd) enabled

Figure 43. Mode selection table

Initialization, continued

The Mode Register is loaded by the fmode operation. This causes the Aadd, Cadd and ABin$_{2..0}$ fields in the instruction word to be loaded into the Mode Register as shown by figure 44. The new mode takes effect at the end of the cycle in which fmode was executed.

C BIT	NORMAL USE	MODE BIT	COMMENT
0	Encn0	0	FPCN is disabled during fmode
1	Encn1	0	
2	Mbin-	0	Mbin- is tied to GND
3	Adst0	1	ALU destination is C bus only
4	Adst1	1	
5	Abin0	M10	
6	Abin1	M11	
7	Abin2	M12	
8	Dadd0	0	
9	Dadd1	0	
10	Dadd2	0	
11	Dadd3	0	
12	Dadd4	0	
13	IOCt0	0	I/O nop specified
14	IOCt1	0	
15	Cwen-	1	C port register writes disabled
16	Cadd0	M5	
17	Cadd1	M6	
18	Cadd2	M7	
19	Cadd3	M8	
20	Cadd4	M9	
21	Badd0	1	fmode function code
22	Badd1	1	
23	Badd2	0	
24	Badd3	0	
25	Badd4	0	
26	Aadd0	M0	
27	Aadd1	M1	
28	Aadd2	M2	
29	Aadd3	M3	
30	Aadd4	M4	
31	F0	0	Miscellaneous function selector
32	F1	0	
33	F2	0	

Figure 44. Load Mode Register instruction format

Initialization, continued

RESET SEQUENCE

Before initializing the contents of the Mode Register, the XL-3232 must be set to a stable state after power up.

Repeating nop and I/O nop instructions for at least four cycles will flush the multiplier/accumulator pipeline and allow the internal states to become well-defined. Until this sequence terminates, the rest of the system should ignore the contents of the data buses and the state of the ZERO, FPCN and FPEX pins.

The registers should then all be initialized to known values (including the Mode, Condition, Status and Tregs) while nops continue to be input. The XL-3232 is then able to begin normal operation.

Division

DIVIDE LOGIC UNIT

The XL-3232 has a divide logic unit. This unit consists of a look-up table ROM and three delay stages. The first cycle of the flut operation transfers the Aadd operand (a) to the divide logic unit. During the next two cycles this operand selects a seed value for the reciprocal of the operand ($1/a$) from the look-up table. This result is written to the Cadd register on the fourth cycle.

The result can be copied to a multiplier/accumulator input port at the same time that the Cadd register is being written.

The look-up result is an IEEE single-precision number whose fraction is accurate to seven bits of precision. If the input is positive or negative infinity (greater than #7F820000 or less than #FF820000), the result is zero. If the input is zero, the result is #7FFFFFFF (which gets clamped to #7F820000 during refinement). flut does not update the Zero, Condition, or Status Registers.

NOTATION:

a = divisor (.f1)

R_0 = seed for $1/a$ (.f31)

R_1 = first approximation (.f31)

R_2 = second approximation (.f30)

b = dividend (.f0)

b/a = result (.f0)

ALGORITHM:

$R_1 = R_0 \times (2 - a \times R_0)$

$R_2 = R_1 \times (2 - a \times R_1)$

Cycle #	Opcode					I/O	Comment
1	flut	.f1,			.f31		R_0 ($\simeq 1/a$)
2	fnop						
3	fnop						
4	fmna	.f1,	.f31,	2,	.f30		$2 - a \times R_0$
5	fnop						
6	fnop						
7	fmac	.f31,	.f30,	0,	.f31		$R_1 = R_0 \times (2 - a \times R_0)$
8	fnop						
9	fnop						
10	fmna	.f1,	.f31,	2,	.f30		$2 - a \times R_1$
11	fnop						
12	fnop						
13	fmac	.f31,	.f30,	0,	.f30		$R_2 = R_1 \times (2 - a \times R_1)$
14	fnop						
15	fnop						
16	fmac	.f0,	.f30,	0,	.f0		$B \times 1/a$
17	fnop						
18	fnop						
19	fnop					; fstore .f0	store b/a
20	fnop						b/a on X port

Figure 45. Recommended division sequence

Division, continued

DIVIDE CODE SUPPORT

The initial approximation to 1/*a* has to be refined by successive approximation. This accurate value of 1/*a* must then be multiplied by *b* in order to complete the divide operation (*b*/*a*).

The programmer or compiler has to supply code to support the following sequence of operations:

1. Execute flut to obtain seed value.

2. Iterate from this value to obtain an accurate divisor inverse using the *Newton-Raphson* algorithm.

3. Multiply the dividend by the inverse of the divisor just generated.

For division, the *Newton–Raphson* algorithm converges quadratically. Theoretically, the number of bits of precision doubles with each iteration. Thus, two iterations should provide the full 23 bits of precision representable by the IEEE 32-bit format. Quantization errors introduced by rounding, however, can prevent the lsb from being accurate. The code example provides *b*/*a* to 22 bits of precision.

Data Format

32-BIT FLOATING-POINT (IEEE STANDARD)

The IEEE standard 32-bit floating-point word divides into three fields: a sign bit, an 8-bit exponent and a 23-bit fraction field (shown in figure 46).

The value contained in the 8-bit exponent field ranges from −127 to 128 (#00 to #FF) (shown in figure 47). The fraction is multiplied by two raised to this power to produce a floating-point value.

The significand field contains the 23-bit fraction and the hidden bit. Inserted during arithmetic processing, the hidden bit has a value of one for all normalized numbers and zero for zero. The fraction is the 23 bits to the right of the hidden bit. Bit F_{22} has a value of 2^{-1}; bit F_0 has a value of 2^{-22}; the hidden bit has a value of 2^0.

All constants are in this IEEE format.

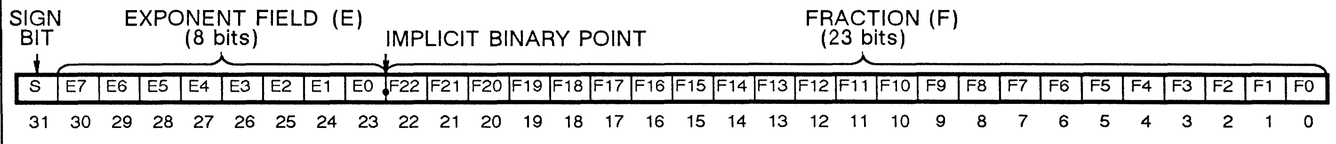


Figure 46. 32-bit floating-point (IEEE standard)

The value of an IEEE floating-point number is determined by the following:

E	F	VALUE	DESCRIPTION
1-254	Any	$(-1)^S\ (1.F)\ 2^{E-127}$	Normalized number (NRN)
0	Any	$(-1)^S\ 0.0$	Zero

Figure 47. 32-bit floating-point value

Data Format, continued

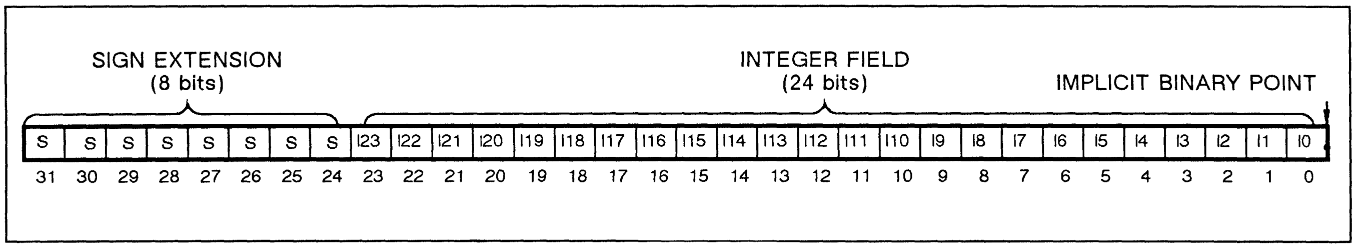


Figure 48. 24-bit fixed point (two's complement)

24-BIT TWO'S COMPLEMENT INTFGERS

The value of the 24-bit integer field shown in figure 48 can range from (-2^{23}) to $(2^{23}-1)$ and must be sign-extended to 32-bits to be compatible with the XL-3232 format. The eight-bit sign extension field is a repeat of bit 23, the sign bit of the two's complement number.

The user must ensure that integer operands conform to this format: integer results are automatically sign-extended to match.

FIX

The fix function converts a number from floating-point format to sign-extended 24-bit two's complement integer format.

If the magnitude of its operand is greater than 2^{22}, Encn1 is set to 1, and M1 = 1, it will set the Condition Register to 1. This limit was chosen to allow software to test for the case $n = 2^{23}$, which cannot be represented by a 24-bit two's complement number. If the operand is too large to be represented in the integer format, the result is clamped to either #007FFFFF or #FF820000, according to its sign.

fix does not attempt to set the Zero or Status Registers. It executes in the same number of cycles as every other multiplier/accumulator instruction.

FLOAT

The float function converts a number from sign-extended two's complement integer format to floating-point format.

If its operand does not have consistent sign extension (bits 24–31 all equal), Encn1 is set to 1 and M1 = 1, it will set the Condition Register to 1. The result of a float operation on such an operand is not defined.

float does not attempt to set the Zero or Status Registers. It executes in the same number of cycles as any other multiplier/accumulator instruction.

IEEE Considerations

The XL-3232 complies with the IEEE Standard for Binary Floating-point Arithmetic (P754) in most respects. The differences described below apply to all of the arithmetic functions (fsubr, fsub, fadd, fmna, fmns, fmac, fabs).

DENORMALIZED NUMBERS

Denormalized numbers have a magnitude less than 2^{-126} but greater than zero. The IEEE standard includes denormalized numbers to allow gradual underflow for operations that produce results that are too small to be expressed as normalized numbers. The XL-3232 do not support denormalized numbers. If the result of an operation is smaller than 2^{-126}, it is replaced by zero and the Zero Register is set to 1. Denormalized operands are detected and flushed to zero (with the same sign) before the operation is performed; no indication of this is provided.

NOT A NUMBER (NAN) HANDLING

The IEEE standard represents NaNs with numbers that have the maximum exponent value and a non-zero fraction. The XL-3232 does not detect attempts to perform calculations on NaNs. Only the flut operation may produce a NaN (when given zero as an operand). This is clamped to the appropriate infinity when refined by the divide code example given on page 37. Other operations may have undefined effects when given a NaN as an operand, so their use should, in general, be avoided. No arithmetic operation generates NaNs; all results with the maximum exponent have their fractions held to zero.

INFINITY AND OVERFLOW

The IEEE standard represents infinities with numbers that have the maximum exponent value and zero as the fraction. The XL-3232 does not detect attempts to perform calculations on infinite operands. Some operations may have undefined effects when given an infinite operand, so their propagation should, in general, be avoided. However, if an operation creates a result that is too large to be represented in the floating-point format, its result is clamped to an infinite value as required by the specification. The Status Register is set by the creation of an infinite value during an operation.

UNDERFLOW

When the result of an operation has a magnitude in the range $0 < n < 2^{-126}$, the XL-3232 rounds it to zero and set the Zero Register to 1. There is no way to distinguish underflow from a result that is exactly zero.

ROUNDING

The XL-3232 supports only the round-to-nearest mode: the infinitely precise result of an operation is rounded to the closest representation that fits in the destination format. If the result is exactly halfway between two representations, it is rounded to the nearest even fraction.

The IEEE standard requires rounding to occur after each arithmetic operation. The XL-3232 does not round between the multiply and add components of the fmac, fmns and fmna functions. The error in the result is always less than two least-significant bits.

If the ABin port of the ALU is set to the constant 0.0, then the fmac function performs a multiply that conforms to the IEEE standard. The fix operation only can be set to round to negative infinity by clearing M1 to zero.

DC Specifications

ABSOLUTE MAXIMUM RATINGS

Supply voltage	−0.5 to 7.0 V
Input voltage	−0.5 V to V_{DD}
Output voltage	−0.5 V to V_{DD}
Operating temperature range (T_{CASE})	−55°C to 125°C
Storage temperature range	−65°C to 150°C
Lead temperature (10 seconds)	300°C
Junction temperature	175°C

RECOMMENDED OPERATING CONDITIONS

PARAMETER		COMMERCIAL		UNIT
		MIN	MAX	
V_{DD}	Supply voltage	4.75	5.25	V
T_{CASE}	Operating temperature	0	85	°C

DC ELECTRICAL CHARACTERISTICS

PARAMETER		TEST CONDITIONS	COMMERCIAL[1]			UNIT
			MIN	TYP	MAX	
V_{IHC}	High level clock input voltage	V_{DD} = MAX	2.4			V
V_{ILC}	Low level clock input voltage	V_{DD} = MIN			0.8	V
V_{IH}	High level input voltage	V_{DD} = MAX	2.0			V
V_{IL}	Low level input voltage	V_{DD} = MIN			0.8	V
V_{OH}	High level output voltage	V_{DD} = MIN, I_{OH} = −1.0 mA	2.4			V
V_{OL}	Low level output voltage	V_{DD} = MIN, I_{OL} = 4.0 mA			0.4	V
I_{IH}	High level input current	V_{DD} = MAX, V_{IN} = V_{DD}			10	μA
I_{IL}	Low level input current	V_{DD} = MAX, V_{IN} = 0V			10	μA
I_{OZL}	Tri-state leakage current low	V_{DD} = MAX, V_{IN} = 0V			10	μA
I_{OZH}	Tri-state leakage current high	V_{DD} = MAX, V_{IN} = V_{DD}			10	μA
I_{DD}	Supply current	V_{DD} = MAX, T_{CY} = MIN TTL inputs[2]			200	mA
C_{IN}	Input capacitance[3]	V_{DD} = 5.0V		10	10	pF
C_{CLK}	Clock capacitance[3]	$T_{AMBIENT}$ = 25°C		25	30	pF
C_{OUT}	I/O, Output capacitance[3]	f = 1 MHz		15	20	pF
C_{OE}	OEX−, OEZ− capacitance[3]			20	25	pF

NOTES: 1 Worst case over power and temperature range.
2 Input levels are 0.4V and 3.4V
3 Capacitances are not tested

Timing Diagrams

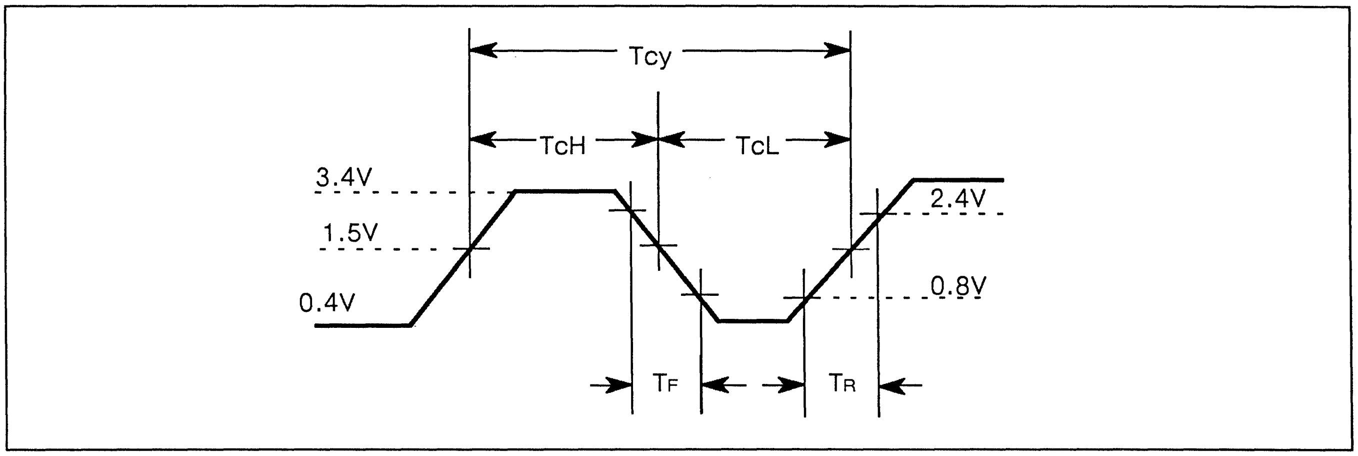
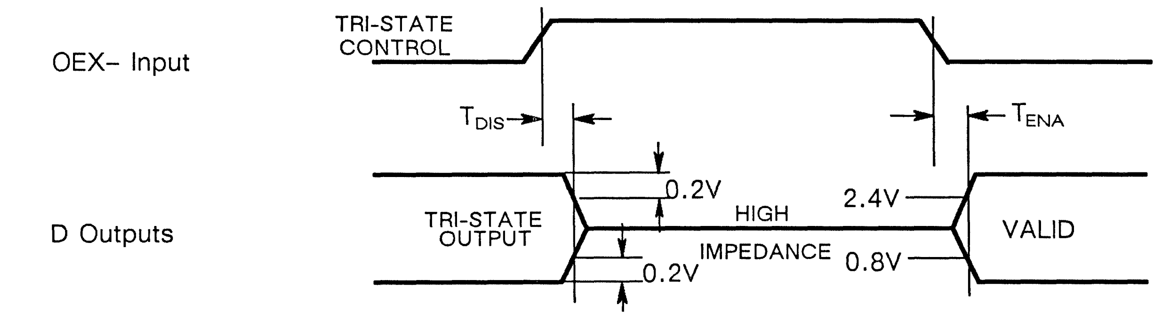


Figure 49. Clock timing



Notes: 1. TTL inputs of 0.4V and 3.5V
2. Timing transitions are measured at 1.5V unless otherwise specified
3. T_{OZ} is not measured but is guaranteed by design

Figure 50. Tri-state timing

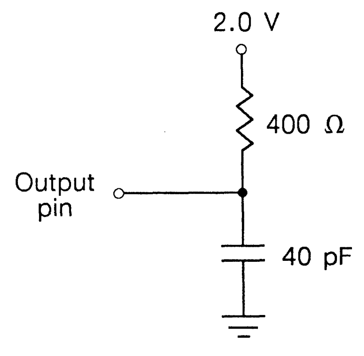


Figure 51. Test load for delay measurement

Timing Diagrams, continued

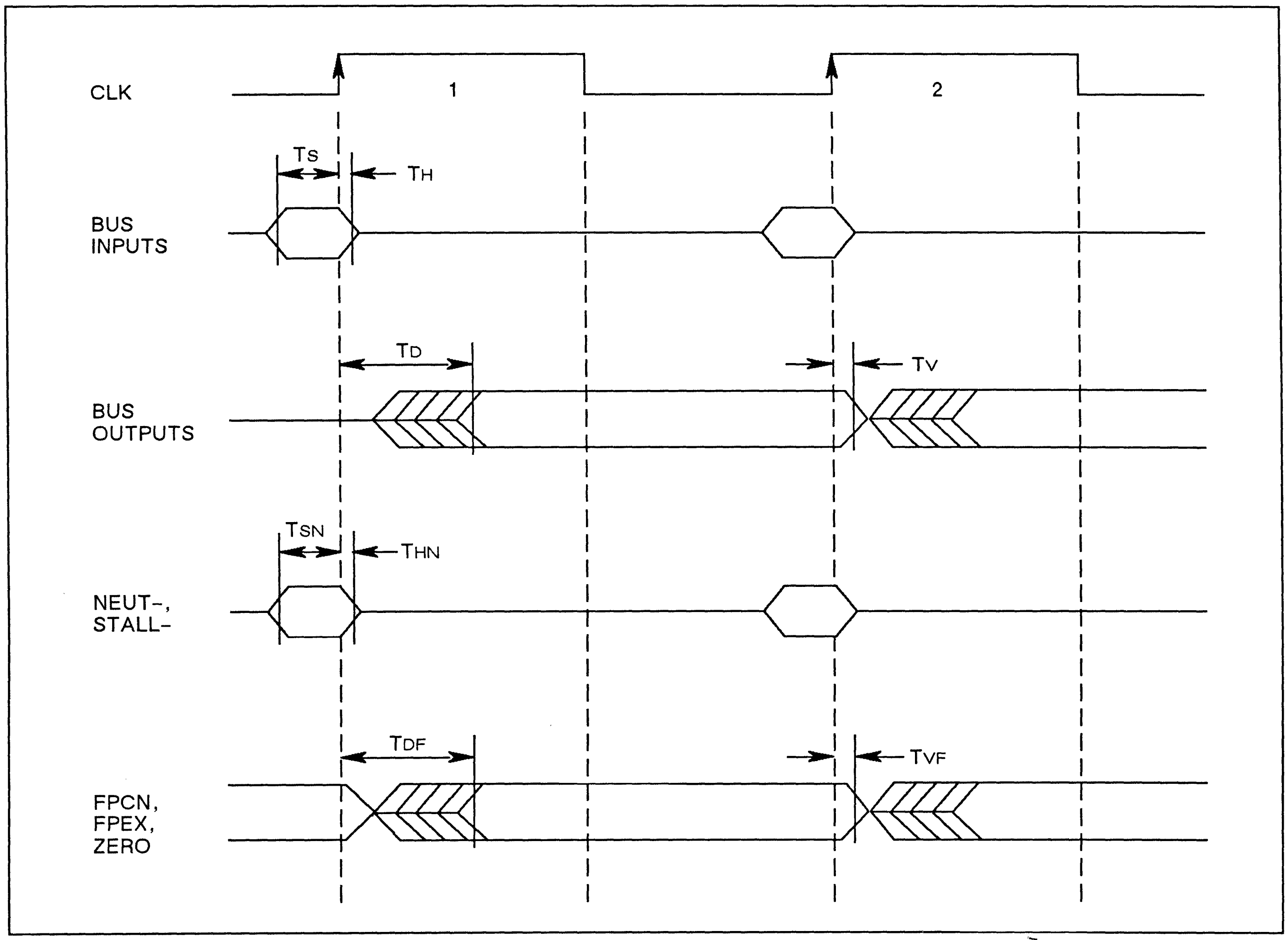


Figure 52. Signal timing diagram

AC Specifications

AC TEST CONDITIONS				
V_{CC} = MIN	V_{IH} = 3.4V V_{IL} = 0.4V	V_{OH} = 2.8V, I_{OH} = -1.0 mA V_{OL} = 0.4V, I_{OL} = 4.0 mA	T_{CASE} = 85 °C	C_{LOAD} = 40 pF

DESCRIPTION		XL-3232-20		XL-3232-40		XL-3232-60		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	
T_{CY}	Clock cycle time	200		120		80		ns
T_{CH}	Clock high time	90		50				ns
T_{CL}	Clock low time	90		50				ns
T_R	Clock rise time				5			ns
T_F	Clock fall time				5			ns
Bus inputs (C, X):								
T_S	Input setup time	30		20				ns
T_H	Input hold time	2		2				ns
Bus outputs (X, Z):								
T_{D0}	Output delay time	3	60	3	35			ns
T_{ENA}	Tri-state enable time		40		35			ns
T_{DIS}	Tri-state disable time[1]		40		35			ns
T_{OP}	Pipelined operation time per stage	200		120		80		ns
T_{LA}	Total latency register-to-register	600		360		240		ns

NOTES: Values shown are at worst-case over the power and temperature range. TTL input levels are 0.4 and 3.4 V. Timing transitions are measured at 1.5 V unless otherwise noted.

The XL-3232 must have power applied for at least 20 ms before initialization and use.

1. T_{DIS} is not tested but is guaranteed by design.

Figure 53. Guaranteed switching characteristics over commercial temperature range and operating conditions

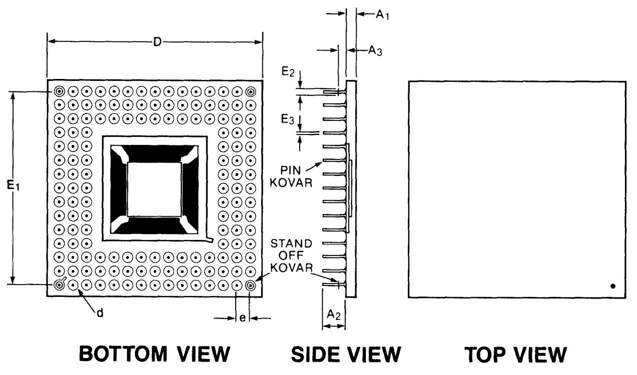
Pin Configuration

Pin #1 Identifier	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	GND	NC	NC	X18	NC	NC	NC	X23	VDD	NC	NC	X27	NC	NC	NC
B	NC	NC	GND	X16	NC	NC	X20	X22	X24	X25	X26	NC	X29	X31	GND
C	NC	NC	X15	GND	X17	X19	X21	NC	VDD	NC	X28	NC	X30	TIE LOW	F2
D	NC	X12	X14										GND	F0	Adst1
E	X10	X11	X13										F1	Adst0	Abin0
F	X8	NC	NC										Abin2	Abin1	GND
G	NC	X9	VDD										STALL-	NEUT-	Cwen-
H	NC	X7	VDD					XL-3232 TOP VIEW					Cadd2	Cadd4	Cadd3
J	X6	X5	NC										Cadd1	Aadd3	Cadd0
K	NC	NC	NC										Aadd1	Aadd2	Aadd4
L	X4	X2	NC										Badd0	Badd4	Aadd0
M	X3	X1	GND										Dadd2	Badd1	Badd3
N	NC	NC	OEX-	VDD	FPCN	GND	TIE LOW	TIE LOW	GND	GND	GND	VDD	Dadd1	Dadd4	Badd2
P	X0	TIE LOW	Encn1	FPEX	Encn0	TIE LOW	CLK	GND	GND	GND	GND	GND	VDD	Dadd0	Dadd3
R	ZERO	GND	IOCt0	IOCt1	Mbin- (TIE LOW)	TIE LOW	TIE LOW	GND	GND	GND	GND	GND	GND	GND	GND

Notes: Pins marked "Tie Low" must be connected to ground. Pins marked "NC" should be left unconnected (floating).

Figure 54. XL-3232 pin configuration

Packaging



Symbol	DIMENSIONS	
	INCHES	MM
A1	0.080 ± 0.008	2.03 ± 0.20
A2	0.180 typ.	4.57 typ.
A3	0.050	1.27
D	1.575 sq. ± 0.016	40.0 sq. ± 0.41
E1	1.400 sq. ± 0.012	35.56 sq. ± 0.30
E2	0.050 dia. typ.	1.27 dia. typ.
E3	0.018 ±0.002	.46 ± 0.05
d	0.070 dia. typ.	1.78 dia. typ.
e	0.100 typ.	2.54 typ.

Figure 55. 144-pin PGA packaging for the XL-3232

Ordering Information

XL-8200 Series customers should order the following chip sets:

SPEED	PACKAGE TYPE	DEVICES	TEMPERATURE RANGE	ORDER NUMBER
-20	2 145-pin plastic PGAs 1 144-pin ceramic PGA	3	T_c = 0-85 °C	XL-8232-020-GPU
-20	2 145-pin ceramic PGAs 1 144-pin ceramic PGA	3	T_c = 0-85 °C	XL-8232-020-GCU
-40	2 145-pin plastic PGAs 1 144-pin ceramic PGA	3	T_c = 0-85 °C	XL-8232-040-GPU
-40	2 145-pin ceramic PGAs 1 144-pin ceramic PGA	3	T_c = 0-85 °C	XL-8232-040-GCU
-60*	2 145-pin plastic PGAs 1 144-pin ceramic PGA	3	T_c = 0-85 °C	XL-8232-060-GPU
-60*	2 145-pin ceramic PGAs 1 144-pin ceramic PGA	3	T_c = 0-85 °C	XL-8232-060-GCU

Figure 56. Ordering information for the XL-8232 (* indicates devices that are not currently available)

Individual XL-3232s can be ordered as follows:

SPEED	PACKAGE TYPE	TEMPERATURE RANGE	ORDER NUMBER
-20	144-pin ceramic PGA	Tc = 0 to +85 ° C	XL-3232-020-GCD
-40	144-pin ceramic PGA	Tc = 0 to +85 ° C	XL-3232-040-GCD
-60*	144-pin ceramic PGA	Tc = 0 to +85 ° C	XL-3232-060-GCD

Figure 57. Ordering information for the XL-3232 (* indicated devices that are not currently available)

XL-3232 32-BIT GRAPHICS FLOATING POINT COMPUTATION UNIT

PRELIMINARY DATA

August 1988

For additional information on WEITEK products, please fill out the form below and mail.

Name ____________________ Title ____________________

Company ____________________ Phone ____________________

Address ____________________

Comments ____________________

I am currently involved in a design with the following Weitek products ____________________ and wish to be added to your design data base to insure that I receive status updates.

APPLICATION:

☐ ENGINEERING WORKSTATIONS ☐ SCIENTIFIC COMPUTERS

☐ GRAPHICS ☐ OTHER ____________________

☐ PERSONAL COMPUTERS

Check the products on which you wish to receive data sheets: ☐ Have a sales person call

ATTACHED PROCESSORS	COPROCESSORS	BUILDING BLOCKS		
☐ XL-SERIES OVERVIEW	☐ 1167	☐ 2264/2265	☐ 1066	☐ 2516
☐ XL-8200 OVERVIEW	☐ 1164/1165	☐ 3132/3332	☐ 2010	☐ 2517
	☐ 3164/3364	☐ 1232/1233	☐ 2245	
	☐ 3167			

WEITEK use: Rec'd Out TPT Source: DS

Status ____________________

WEITEK XL-3232
Please Comment On The Quality Of This Data Sheet.

Even though we have tried to make this data sheet as complete as possible, it is conceivable that we have missed something that may be important to you. If you believe this is the case, please describe what the missing information is, and we will consider including it in the next printing of the data sheet.

Fold, Staple and Mail to Weitek Corp.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1374 SUNNYVALE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

WEITEK Corporation
1060 E. Arques Ave.
Sunnyvale, CA 94086-BRM-9759

ATTN: Ed Masuda

WEITEK

WEITEK'S CUSTOMER COMMITMENT:

Weitek's mission is simple: to provide you with VLSI solutions to solve your compute-intensive problems. We translate that mission into the following corporate objectives:

1. To be first to market with performance breakthroughs, allowing you to develop and market systems at the edge of your art.

2. To understand your product, technology, and market needs, so that we can develop Weitek products and corporate plans that will help you succeed.

3. To price our products based on the fair value they represent to you, our customers.

4. To invest far in excess of the industry average in Research and Development, giving you the latest products through technological innovation.

5. To invest far in excess of the industry average in Selling, Marketing, and Technical Applications Support, in order to provide you with service and support unmatched in the industry.

6. To serve as a reliable, resourceful, and quality business partner to our customers.

These are our objectives. We're committed to making them happen. If you have comments or suggestions on how we can do more for you, please don't hesitate to contact us.

[illegible]

Art Collmeyer
President

Headquarters
Weitek Corporation
1060 E. Arques Avenue
Sunnyvale, CA 94086
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185
TEL (408) 738-8400

Domestic Sales Offices
Weitek Corporation
1060 E. Arques Avenue
Sunnyvale, CA 94086
TWX 910-339-9545
WEITEK SVL
FAX (408) 738-1185
TEL (408) 738-8400

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

European Sales Headquarters
Greyhound House, 23/24 George St.
Richmond, Surrey, TW9 1JY
England
TELEX 928940 RICHBI G
FAX 011-441 940 6208
TEL 011-441 549 0164

Japanese Representative
C. Itoh Techno/Sciences
Company Ltd.
C. Itoh Building
2-5-1 Kita-Aoyama
Minato-Ku, Tokyo 107
TELEX 781242 3240
FAX (81) 3-497-4879
TEL (81) 3-497-4975