# VMECC Specification

Ron Kolb

*Revision 0.97*

# *Table of Contents*

# *List of Tables*

# *List of Figures*

CHAPTER 1

# Introduction

## 1.1 Purpose

The VMECC provides an interface to a VME bus for the Everest system. It is connected to an F-chip on an IO4 or similar board by a 60 signal flat cable. The cable arrangement, even when constrained to be in the same cabinet as the IO4 board and limited to five feet of cable, gives much more flexibility to the placement of the VME bus and allows multiple buses to exist.

## 1.2 Features

The VMECC features include:

- A16, A24, A32, A64 addressing modes can be issued as bus master with PIOs.
- Single item transfers (D8, D16, D32, and D64) can be issued as bus master with PIOs.
- A24, A32, A64 addressing modes can be responded to as a memory slave providing access to the EBUS; A24 and A32 access is mapped on a page basis; A64 access is direct (unmapped) to EBUS addresses.

- Single item transfers (D8, D16, D32), Block Transfer (D8, D16, and D32), and Multiplexed Transfer (D64) can be responded to as a memory slave providing access to the EBUS.
- The VMECC includes an internal DMA engine to speed copies between EBUS memory and VME space. The addressing mode bits, VME address, datum size, block/non-block, a bus throttle value, direction, and transfer size are all programmable. The VME address is a single incrementing range of addresses; the EBUS address is likewise a single incrementing range, but because it is mapped, the physical EBUS address can scatter/gather on a page basis.
- The VMECC has a 16-deep PIO fifo to smooth writing to the VME bus from R4000s.
- The VMECC has an explicit internal delay register to aid in spacing PIOs for boards that require such a kludge.
- The VMECC has provision for Test and Set operations for D8, D16, D32 with A16, A24, and A32 on the VME bus using a VME bus Read Modify Write cycle.
- The VMECC has a built-in VME interrupt handler. It will be possible for an external agent to handle some of the interrupt levels.
- The VMECC has a built-in VME bus arbiter.
- VME “slot 0” functions such as bus timer, iack driver, clock driver, etc. are supported in the VMECC.
- The VMECC has two general purpose input pins that can cause EBUS interrupts and two general purpose output pins.

# CHAPTER 2 Software Specification

## 2.1 VME PIO Addressing

### 2.1.1 Little Windows and Big Windows

Each IOA (the F-chip in this case) on the I-bus has 64KB of little window EBUS address space that can be accessed without TLB entries and 128MB of big window address space that does require TLBs. The F-chip combines the two spaces into one 128MB piece (the first 64KB of the big window accesses the same locations as the little window) and takes the first 4KB for its own internal registers. The rest (128MB minus 4KB) is passed to the VMECC.

The VMECC divides the 128MB into 16 8MB pieces. The first 8MB piece is for internal registers and A16 space. The VMECC has a 15 entry mapping ram for pieces 1-15. Each 8MB piece is expanded to 32 bits of address with 9 bits from the ram and given a 6 bit AM field. There is a single 32 bit register to provide the additional bits for A64. It is shared by both PIOs and the DMA engine.

PIO reads and writes of double word (8byte) size to the VME bus have the lower two AM bits forced to 0b00 (D64) to conserve mapping entries. If this were not done, two entries would be needed for a given address extension, one with the AM bits set to Axx/D64 and one with the AM bits set to Axx with some other pattern in the two least significant bits.

**Table 1: PIO Big Window Mapping**

| Address Range | | Used For |
|---|---|---|
| 0000000 | 0000FFF | F-chip addresses (VMECC won't see this range) |
| 0001000 | 0001FFF | VMECC internal registers |
| 0002000 | 0003FFF | RMW trigger |
| 0004000 | 0007FFF | A16 SP (AM=2D) addresses 0000 - 3FFF |
| 0008000 | 000BFFF | A16 NP (AM=29) addresses 8000 - BFFF |
| 000C000 | 000FFFF | A16 SP (AM=2D) addresses 8000 - BFFF |
| 0010000 | 001FFFF | A16 NP (AM=29) addresses 0000 - FFFF |
| 0020000 | 002FFFF | A16 SP (AM=2D) addresses 0000 - FFFF |
| 0030000 | 07FFFFF | reserved (actually aliases for previous addresses) |
| 0800000 | 0FFFFFF | mapper entry #1 |
| 1000000 | 17FFFFF | mapper entry #2 |
| 1800000 | 1FFFFFF | mapper entry #3 |
| 2000000 | 27FFFFF | mapper entry #4 |
| 2800000 | 2FFFFFF | mapper entry #5 |
| 3000000 | 37FFFFF | mapper entry #6 |
| 3800000 | 3FFFFFF | mapper entry #7 |
| 4000000 | 47FFFFF | mapper entry #8 |
| 4800000 | 4FFFFFF | mapper entry #9 |
| 5000000 | 57FFFFF | mapper entry #10 |
| 5800000 | 5FFFFFF | mapper entry #11 |
| 6000000 | 67FFFFF | mapper entry #12 |
| 6800000 | 6FFFFFF | mapper entry #13 |
| 7000000 | 77FFFFF | mapper entry #14 |
| 7800000 | 7FFFFFF | mapper entry #15 |

**Table 2: Internal Register Summary**

| Register Name | Address |
|---|---|
| CONFIG | 1000 |
| A64SLVMATCH | 1008 |
| A64MASTER | 1010 |
| ERRADDRVME | 1018 |
| RMWMASK | 1080 |
| RMWSET | 1088 |
| RMWADDR | 1090 |
| RMWAM | 1098 |
| DMAVADDR | 1100 |
| DMAEADDR | 1108 |
| DMABCNT | 1110 |
| DMAPARMS | 1118 |
| PIOTIMER | 1180 |
| LOOPBACKTRIG | 11C0 |
| INT_ENABLE | 1200 |
| INT_REQUESTSM | 1208 |
| INT_REQUESTRAW | 1210 |
| ERRCAUSES | 1218 |
| INT_ENABLECLR | 1220 |
| ERRCAUSESCLR | 1238 |
| INT_ENABLESET | 1240 |
| ERRXTRAVME | 1280 |
| IACK | 1288-12B8 |
| VECTORS | 1300-1350 |
| PIOMAP | 1380-13F8 |
| RMWTRIG | 2000-2007 |

### 2.1.2 RMW operation

There are five registers associated with performing the Test and Set operation on the VME bus with a RMW operation. One 32 bit register is the address and another 6 bit register is for the AM bits. Only A16, A24 and A32 addresses can be used (no A64) and D64 is not supported. Two more 32 bit registers provide a mask and a set value. All of these registers must be written before initiating the operation with a read to the trigger location. The size of the operation (byte, half-word, word) is determined by the size of the read at the trigger location. Once the RMW logic has the VME bus, it performs a VME read of the requested data size with the address and AM bits specified. This is the data that is returned from the trigger read over the FCI. However, before any other PIOs in the fifo can be started, a copy of the read data is ANDed with the mask value and ORed with the set value. It is written back to the same VME location that was read. The VME AS signal remains asserted during both the VME read and write operations. Once the VME write finishes, the VMECC may proceed with any further PIOs in its fifo. Since there are five distinct registers to be written and read, software must protect access to them with a software lock. To simplify the hardware, the values in the AND and OR registers must be duplicated in all four bytes for byte operations and duplicated in both half-word locations for half-word operations.

While the RMWAM register can be written to any value (and that value will be returned by a read to its address), the AM pattern which is actually placed on the VME bus will have the two least significant bits set to 0b01 (data). For the anticipated use of this set and set mechanism, neither program, block, or D64-block AM codes make sense.

The RMWTRIG address used must match the value in the RMWADDR location in the three least significant bits. For example, if the value in the RMWADDR register is 0x1234567B, the RMWTRIG trigger read should be from address 0x2005.

## 2.2 VME Slave Addressing

### 2.2.1 A24, A32, and A64 Enable Bits

There is a configuration register to enable and disable sections of A24, A32, and A64 address space being recognized as valid slave addresses. The IO2/IO3 recognized all A24 NP or Block accesses with bit A23

equal to zero, and, as usually configured, all A32 NP and Block accesses with the A31-A28 equal to 1000B or 0000B. This behavior can be imitated in the Everest system, although a better partitioning is suggested. An option bit allows access to SP as well as NP address modifiers.

On Everest, A24 will be divided into four sections decoded from the A23 and A22 bits, and A32 will be divided into 16 sections decoded from the A31 to A28 bits. All A32 and A24 accesses must be mapped. The 2048 static ram entries will be indexed by A31-A21, and A20-A12 are the table index bits within the EBUS memory tables.Which 2048 of the 8192 static ram entries connected to the IA/ID chips will be used is set by two bits in the CONFIG register. For A24 access, the static ram addressing bits A31-A24 will be forced to ones. If the enable matching 1111B for A31-A28 of the A32 enables is turned on, then a part of that part of A32 space's mapping entries will be also used by the A24 enables (like the IO2/3).

The IO2/IO3 A24 enable pattern of 00B/01B turned on and 10B/11B turned off can be programmed, but for co-existence with the possibility of another CPU and local memory on the VME bus, it might be better to have 01B/10B turned on and 00B/11B turned off. This allows the bottom and top of the address space for local VME memory. Similarly, for A32, turning on 1000B to 1110B with 1111B turned off might be better.

There is a single bit in the CONFIG register to enable or disable A64 slave operations.

### 2.2.2 A64 Upper Match Register

The VMECC is capable of responding to one 2**40 bit section of A64 space. There is a configuration register to set the pattern of A63-A40 to match to respond to A64 addresses as a slave.

## 2.3 Interrupts

### 2.3.1 VME levels

The seven VME levels have an EBUS interrupt vector each. These can be reprogrammed by PIO writes at any time and any interrupts occurring as the switch over happens are guaranteed to have either the old or new values.

The VMECC has a mask register with a bit for each of the seven VME levels. If a VME interrupt level is asserted and the mask bit is enabled, the built-in interrupt handler will request the VME bus and do the IACK cycle automatically. The eight bit VME IACK vector value will be stored in a register file location associated with its IRQ level. The mask bit is automatically turned to the disabled state, so that further IACKs and interrupts from that level won't happen until the level is turned on by a PIO write from an R4000. This prevents a continuous series of IACKs for boards that don't remove the IRQ when the IACK happens and also preserves the IACK vector from being overwritten before examination by the R4000.

The mask register bits can be read, cleared with a bit mask, and set with a bit mask for easier manipulation by the R4000s.

### 2.3.2 Auxiliary Interrupt inputs

There are two pins on the VMECC that cause interrupts back to the EBUS when they are asserted. They have two additional EBUS interrupt vector values and two additional mask bits similar to the seven VME level bits. They are active high levels and cause an interrupt on the leading edge; anything fancier can be handled by a PAL on the VCAM.

### 2.3.3 DMA Engine Interrupts

There is an EBUS interrupt vector register for the DMA engine's use. It, too, has a mask bit.

### 2.3.4 Error Interrupts

VMECC detected errors such as Flat Cable Interface protocol errors, VME bus timeouts, etc., share a single EBUS interrupt vector. The conditions have sticky bits in an error register, but the same CPU or CPU group is interrupted for all.

### 2.3.5 General Interrupt Issues

All of the EBUS interrupt vectors must be unique because the VMECC does not have a parallel set of settable and clearable bits which would duplicate the function of the 128 level bits in the CCs. If two VMECC levels were to operate on the same one of the 128 CC levels, a CPU would

have no way to know, for instance, which of the VME IACK RAM locations to look at.

Unlike the MPBUS systems, it is not anticipated that VME interrupt enable bits will be cleared by software in normal operation; they must be set to re-enable interrupts after the auto-clear caused by sending an EBUS interrupt request and they may be cleared permanently to allow another CPU on the VME bus to handle certain VME interrupt levels. If a CPU did clear the interrupt enable bit, interrupts could occur for a basically indeterminate amount of time after issuing the clear. There is no status available to guarantee that an "in-flight" interrupt has reached the target CC.

### 2.3.6 Interrupt Hardware Details

The seven VME interrupt lines are synchronized and *AND*ed with the appropriate INTENABLE bits. Whenever any of these seven terms is set, the VME interrupt handler makes a request to gain control of the VME bus to the VME BUS ARBITER. When it gets control, it issues an IACKOUT and performs an IACK read to the highest priority (1 highest; 7 lowest) VME level. This causes:

- the resulting 8 bit IACK vector to be written into the appropriate IACKVME***x*** register
- the corresponding ***x*** bit in the INTENABLE register to be automatically reset
- the corresponding ***x*** bit in the INTREQUESTSM register to be set

The INTREQUESTSM register can also have bits set by diagnostics software. Whenever any of its bits are set, another state machine will schedule sending interrupt vectors obtained from the VECTOR***xxx*** registers over the FCI bus. It will wait for appropriate conditions such as DMA Write Buffer Flushing, etc. This set of state machine bits has four additional bits beyond the seven VME interrupt levels for the DMA engine, the two auxiliary interrupt inputs, and the error register. For all levels, as the interrupt order is sent over the FCI, the corresponding INTREQUESTSM state machine bit is reset.

The seven VME interrupt levels have their INTENABLE bit reset when an interrupt occurs because there is a need to preserve the contents of the IACKVME***x*** register until software can read it. After software reads the IACKVME***x*** register, it should write to set the ***xth*** bit of the INTENABLE register. A second reason for this auto-reset of an enable bit is that

the VME interrupt levels are level-sensitive (as opposed to edge-triggered). Without the auto-reset, the interrupt handlers might continuously generate VME interrupts if the controller board doesn't remove its IRQ line assertion automatically when it returns the IACK vector. The two auxiliary interrupt pins are not so protected but cause an interrupt only on their leading edge.

The DMA engine and error interrupt INTENABLE bits do not auto-reset; they usually would always be enabled. By its nature, the DMA engine interrupt occurs as the byte count goes to zero (or there's some kind of error) and, since it won't start without some software programming, there is no chance of continuously generating interrupts. The error register case is special. The strategy is that:

- as the number of error bits turned on in ERRCAUSES goes from zero to one or more, an interrupt is sent (the bit in the INTREQUESTSM is set)
- more conditions can occur before an ERRCAUSECLR read is issued (which auto-clears any bits read as ones), but these additional bits don't cause additional interrupts to be sent nor are the bits individually logged. Instead, they cause an overrun bit to be set.

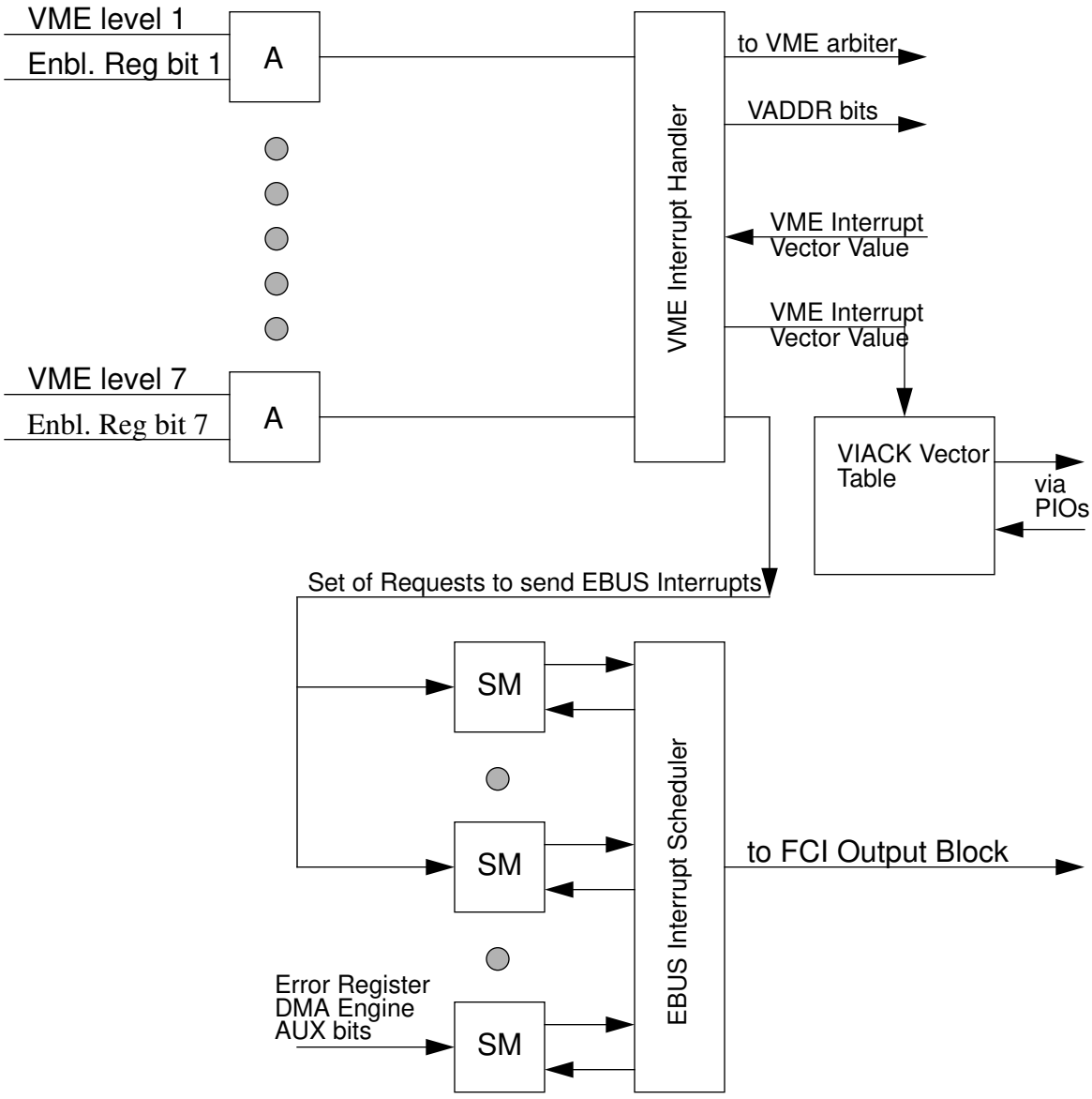


**Figure 1** Interrupt State Machine Concept

## 2.4 Coherence Issues

One of the features of the VMECC is that it will attempt to coalesce VME bus operations that are in the same direction, at adjacent sequential address, with the same datum size, with the same AM bits, and without intervening interrupts. This coalescing in the VMECC is in addition to the coalescing that the F-chip attempts to do with write data sent to it. A timer, with a period on the order of 5 usec, will also break this coalescing. This feature will allow non-block mode devices to access sequential addresses without quite as much overhead as would be required to re-fetch or store every access over the FCI and through the IBUS to the EBUS. It will also help smooth even block mode devices as they must release AS at every 256 byte boundary. Obviously, this requires a “last address plus datum size” register that is compared to new requests.

Pre-fetching of DMA read data can occur if it is properly discarded when a stream is broken. Because pre-fetching to the next page can occur, the map entry beyond the address range actually expected to be used must initialized and allocated to the current request.

One problem that can arise is that the VME master can switch direction from write to read in a given block of data and quite reasonably expects the new values to appear. The write-back of the data must happen before the read request is issued in the VME chip and the F-chip must similarly be aware of this case and make sure that at least the IA sees the write before issuing the read.

Another subtler issue that cannot be handled in the VMECC is that of multiple virtual aliases for a given piece of memory. That is, since all A24 and A32 accesses are mapped, very different VME addresses could refer to the same physical address. The F-chip, since it knows physical addresses, could handle this and make sure that a read sees the effect of a previous write.

All PIOs, both to internal and VME destinations, are handled strictly in order. Pending DMA writes must be flushed onto the FCI before interrupts are posted. The F-chip must preserve that order. As mentioned elsewhere, re-targeting interrupt levels must result in either the old or new CPU getting the interrupt.

## 2.5 Local Register Description and Address Map

The software accessible registers within the VMECC will be described in terms of a number of groups of related registers.

Note that the addresses for the internal registers are appropriate for double word (64 bit) access even though the registers are really only a word (32 bits) or less wide. If accessed as double words, code accessing these registers would be endian insensitive. If it is desired that the registers be accessed with word loads and stores instead, the addresses will have to be modified by adding 4 (bytes) to the addresses in big endian mode; the addresses are correct in little endian. There are no byte enables within words, so one cannot, e.g., write only a byte of a word-wide register and a register that happens to really only be a half-word or byte wide would need to have the address listed in this document adjusted by 6 or 7 (bytes) in big endian mode to position the data into the correct byte lane.

### 2.5.1 RMW Group and Loopback

The RMW operation is a Test and Set type operation and requires a number of registers. The Mask used for the AND function and the Set value used for the OR function are word wide registers. The 6 bit AM register and 32 bit address register are required because the trigger location is a fixed internal address that is read to start the RMW operation. The size of the PIO read operation (byte, half-word, or word only) to the RMWTRIG location is used as the size of the operation on the VME side. Note that A64 and D64 operations are not supported; do not program the RMWAM register to an A64 code. The two least significant AM bits that go out on the VME bus are forced to 0b01 (data). Note also that the three least significant bits of the RMWADDR value and the three least significant bits of the address used for the RMWTRIG access must match for correct operation. Finally, note that the patterns in RMWMASK and RMWSET registers should be repeated twice for half-word operations and repeated in all four byte lanes of the two registers for byte operations.

The FCI loopback trigger uses the low byte of the RWM register for data to check the AUXCMD bits of the FCI.

**Table 3: RMW Group**

| Address | Register Name | Bits | R/W | Function |
|---|---|---|---|---|
| 0001080 | RMWMASK | 31:0 | R/W | *AND*-function for RMW operation |
| 0001088 | RMWSET | 31:0 | R/W | *OR*-function for RMW operation |
| 0001090 | RMWADDR | 31:0 | R/W | Address Bits for RMW operation |
| 0001098 | RMWAM | 5:0 | R/W | AM bits for RMW operation |
| 0002000-0002007 (3 lsb must match RM WADDR) | RMWTRIG | Var. | R | A read from this location triggers RMW operation and returns the contents of AM:ADDR with the operation size obtained from the size of read to this location |
| 00011C0 | LOOPBACKTRIG | NA | W | A write to this location initiates an FCI loopback test with bits 7:0 of the RMWMASK register used as data |

### 2.5.2 Error Group

One register holds the general error cause bits. At one address, it can be read without changing the error bit(s); at another, the error bits that are set in the read data are automatically cleared. Another two registers hold the last VME address along with AM bits, IACK,WRITE, LWORD, DS0, DS1, AS and grant level values along with bits indicating the validity of the two registers.

The first register is the ERRCAUSES register which has a lockout bit that prevents more than one of bits 0, 1, 3, 4, or 5 being set; instead, an error condition different from the first condition to set a bit will cause bit 6, the overrun bit, to be set. Bit 2 (VME slave buserror) does not set the lockout bit and bit 7 (dropmode) is just a “live” status bit.

The ERRXTRAVME register is triggered just before a VME bus timeout asserts BERR. This register does not capture data when a slave asserts BERR, only BERR caused by the bus timer. The ERRADDRVME register is loaded for the addresses the VMECC drives as master and the starting address of slave operations the VMECC handles. When bit 16 of the ERRXTRAVME register (which is the validity bit and also the lockout

bit) sets, bit 15 captures the information that the VMECC is either the master or slave for this operation which indicates whether the ERRADDRVME register's contents apply to this timeout.

The one vector associated with the errors are described in the vector group.

| Address | Register Name | Bits | R/W | Function |
|---|---|---|---|---|
| 0001018 | ERRADDRVME | 31:0 | R | Last VME address (if known), bit 0 is ~LWD |
| 0001280 | ERRXTRAVME | 5:0 | R | Last VME AM bits |
| | | 6 | R | Last VME IACK bit |
| | | 7 | R | Last VME Write bit |
| | | 11:8 | R | Last VME AS:DS1:DS0:LWORD |
| | | 14:12 | R | Last VME Grant Level (6:Interrupt Master; 5:PIO Master: 4: DMA Engine; 3-0 VME backplane levels) |
| | | 15 | R | ERRADDRVME is valid |
| | | 16 | R/W | ERRXTRAVME is valid (cleared and re-armed by writing to this address -- 1280 |
| 0001218 | ERRCAUSES | 0 | R | VME Buserror on PIO Write (interrupts) |
| | | 1 | R | VME Buserror on PIO Read (no interrupt but read return has parity error) |
| | | 2 | R | VME Slave Got Parity Error (no interrupt) |
| | | 3 | R | VME Acquisition Timeout by PIO Master (interrupts and sets drop mode) |
| | | 4 | R | FCIDB Timeout (no interrupt) |
| | | 5 | R | FCI PIO Parity Error (interrupts) |
| | | 6 | R | Overrun Bit (while one of bits 0-1-3-4-5 is set, a different one of bits 0-1-3-4-5 is requested) |
| | | 7 | R | Dropmode is set |
| | | 11:8 | R | Reserved |
| | | 15:12 | R | Chip Revision. Currently 0x0. |
| 0001238 | ERRCAUSECLR | 7-0 | R-C | Same bits as ERRCAUSES with auto-clear |

### 2.5.3 DMA Engine

The DMA engine has a 32 bit EBUS memory address register (DMAEADDR). The address is always virtual (mapped). The engine has a VME address and AM bits register, the DMAVADDR register. The length of the operation is contained in the DMABCNT register. The DMAMODE register has the direction, size of VME bus operation, block/non-block operation, VME AM bits, VME bus throttle value, VME bus request level, and DMA start bit in it. The DMA engine busy bit is set when the DMAMODE location is written with the start bit set and cleared when either the transfer has finished or there is an error. The EBUS interrupt vector associated with it is described in the vector group.

**Table 4: DMA Engine Group**

| Address | Register Name | Bits | R/W | Function |
|---|---|---|---|---|
| 0001100 | DMAVADDR | 31:0 | R/W | DMA engine VME address |
| 0001108 | DMAEADDR | 31:0 | R/W | DMA engine EBUS address |
| 0001110 | DMABCNT | 23:0 | R/W | DMA engine transfer byte count |
| 0001118 | DMAPARMS | 5:0 | R/W | DMA engine AM bits for DMAVADDR |
| | | 6 | R/W | DMA engine should use block/non-block VME transfers |
| | | 8:7 | R/W | DMA engine's VME transfers datum size is: 00-byte, 01-hw, 10-wd, 11-64bit |
| | | 9 | R/W | DMA engine transfers: 0-VME to EBUS; 1- EBUS to VME |
| | | 10 | R/W | DMA VME side is RWD/ROR (at throttle value) |
| | | 11 | R/W | DMA throttle value is 2048/256 bytes |
| | | 29 | R/C | DMA engine got a FCI parity error; cleared by setting bit 31 |
| | | 30 | R/C | DMA engine got a VME buserror; cleared by setting bit 31 |
| | | 31 | R/W | DMA engine is active (set to start xfer) |

When a DMA engine interrupt occurs, software should read the DMAPARMS register to see whether the completion was normal or whether

the operation ended because of a VME bus error or a flat cable parity error.

Like double word PIOs, the DMA engine AM is automatically set to D64 when the size is double word.

### 2.5.4 Configuration Group

The registers in this group are likely to have the same values for all standard systems and certainly only be written at initialization. The parameters are configurable primarily to allow for customer specials. One 32 bit register provides an additional 32 address bits for A64 master operation. A 24 bit register must have its value match A64 address bits A63-40 to allow slave access to EBUS memory. A general configuration register has a variety of configuration bits including which parts of A24 and A32 space and whether A64 space will be enabled for slave access. The other bits are described in the table..

.

**Table 5: Configuration Group**

| Address | Register Name | Bits | R/W | Function |
|---|---|---|---|---|
| 0001000 | CONFIG | 0 | R/W | VME reset asserted when this bit a 0. |
| | | 1 | R/W | PIO master doesn't hold its bus grant for about 5 usec before releasing it even if no VME PIOs are immediately available |
| | | 2 | R/W | DMA engine is VME request level 2.5 (between 3 and 2-1-0) instead of 4 (above 3) |
| | | 4:3 | R/W | Upper bits of Virtual Address on FCI |
| | | 5 | R/W | VME slaves will respond to Privileged AM patterns |
| | | 6 | R/W | Disable VME slave operation coalescing |
| | | 7 | R/W | FCI Command Overlap Enable |
| | | 8 | R/W | VME Bus Timeout value is 640 usec/80 usec |
| | | 9 | R/W | Shorten VME bus timeout and PIO VME bus acquisition access timeout (for diagnostics only) |
| | | 10 | R/W | Reserved |
| | | 11 | R/W | Enable A64 slave operations |
| | | 15:12 | R/W | Each bit is a slave enable for 1/4th of A24 |
| | | 31:16 | R/W | Each bit is a slave enable for 1/16th of A32 |
| 0001008 | A64SLVMATCH | 31:8 | R/W | A64 Slave Match Address (VMEA63-40) |
| 0001010 | A64MASTER | 31:0 | R/W | A64 PIO Master/DMA Engine upper 32 bits |

### 2.5.5 Interrupts, IACK Values, and General Purpose I/O

This group contains all of the EBUS interrupt vectors in the VMECC, the corresponding interrupt mask bits, the 7 VME IACK values associated with the 7 VME IRQ levels, and a two general I/O bits for attached processor use. There are four EBUS interrupt vectors in addition to the 7 VME IRQ levels. One is used to signal general VMECC errors, both on the FCI and on the VME bus. Another signals the end of a DMA engine operation. Two general input pins cause interrupts when asserted (high

true levels). They can be used for the generic external interrupt lines often requested or associated with an attached processor. Two general purpose output pins are part of the interrupt mask register and can be set and cleared with the mask clear and mask set operations but do not actually sense or cause interrupts

.

**Table 6: Interrupt Vector Group**

| Address | Register Name | Bits | R/W | Function |
|---|---|---|---|---|
| 0001300 | VECTORERROR | 15:0 | R/W | Interrupt Vector for VMECC errors |
| 0001308 | VECTORVME1 | 15:0 | R/W | Interrupt Vector for VME level 1 |
| 0001310 | VECTORVME2 | 15:0 | R/W | Interrupt Vector for VME level 2 |
| 0001318 | VECTORVME3 | 15:0 | R/W | Interrupt Vector for VME level 3 |
| 0001320 | VECTORVME4 | 15:0 | R/W | Interrupt Vector for VME level 4 |
| 0001328 | VECTORVME5 | 15:0 | R/W | Interrupt Vector for VME level 5 |
| 0001330 | VECTORVME6 | 15:0 | R/W | Interrupt Vector for VME level 6 |
| 0001338 | VECTORVME7 | 15:0 | R/W | Interrupt Vector for VME level 7 |
| 0001340 | VECTORDMAEN | 15:0 | R/W | Interrupt Vector for DMA Engine |
| 0001348 | VECTORAUX0 | 15:0 | R/W | Interrupt Vector for general purpose intr. 0 |
| 0001350 | VECTORAUX1 | 15:0 | R/W | Interrupt Vector for general purpose intr. 1 |
| 0001288 | IACK1 | 7:0 | R/W | VME IACK vector for VME level 1 |
| 0001290 | IACK2 | 7:0 | R/W | VME IACK vector for VME level 2 |
| 0001298 | IACK3 | 7:0 | R/W | VME IACK vector for VME level 3 |
| 00012A0 | IACK4 | 7:0 | R/W | VME IACK vector for VME level 4 |
| 00012A8 | IACK5 | 7:0 | R/W | VME IACK vector for VME level 5 |
| 00012B0 | IACK6 | 7:0 | R/W | VME IACK vector for VME level 6 |
| 00012B8 | IACK7 | 7:0 | R/W | VME IACK vector for VME level 7 |

**Table 7: Interrupt Mask Group**

| Address | Register Name | Bits | R/W | Function |
|---|---|---|---|---|
| 0001200 | INT_ENABLE | 0 | R/W | Enable bit for VMECC errors |
| | | 1 | R/W | Enable bit for VME IRQ level 1 |
| | | 2 | R/W | Enable bit for VME IRQ level 2 |
| | | 3 | R/W | Enable bit for VME IRQ level 3 |
| | | 4 | R/W | Enable bit for VME IRQ level 4 |
| | | 5 | R/W | Enable bit for VME IRQ level 5 |
| | | 6 | R/W | Enable bit for VME IRQ level 6 |
| | | 7 | R/W | Enable bit for VME IRQ level 7 |
| | | 8 | R/W | Enable bit for DMA engine |
| | | 9 | R/W | Enable bit for general purpose input 0 |
| | | 10 | R/W | Enable bit for general purpose input 1 |
| | | 11 | R/W | General purpose output bit 0 |
| | | 12 | R/W | General purpose output bit 1 |
| 0001208 | INT_REQUESTSM | 0 | R/W | Interrupt Request for VMECC errors |
| | | 1 | R/W | Interrupt Request for VME IRQ level 1 |
| | | 2 | R/W | Interrupt Request for VME IRQ level 2 |
| | | 3 | R/W | Interrupt Request for VME IRQ level 3 |
| | | 4 | R/W | Interrupt Request for VME IRQ level 4 |
| | | 5 | R/W | Interrupt Request for VME IRQ level 5 |
| | | 6 | R/W | Interrupt Request for VME IRQ level 6 |
| | | 7 | R/W | Interrupt Request for VME IRQ level 7 |
| | | 8 | R/W | Interrupt Request for DMA engine |
| | | 9 | R/W | Interrupt Request for general purpose input 0 |
| | | 10 | R/W | Interrupt Request for general purpose input 1 |

**Table 7: Interrupt Mask Group (Continued)**

| Address | Register Name | Bits | R/W | Function |
|---|---|---|---|---|
| 0001240 | INT_ENABLESET | 0 | S | Enable bit set for VMECC errors |
| | | 1 | S | Enable bit set for VME IRQ level 1 |
| | | 2 | S | Enable bit set for VME IRQ level 2 |
| | | 3 | S | Enable bit set for VME IRQ level 3 |
| | | 4 | S | Enable bit set for VME IRQ level 4 |
| | | 5 | S | Enable bit set for VME IRQ level 5 |
| | | 6 | S | Enable bit set for VME IRQ level 6 |
| | | 7 | S | Enable bit set for VME IRQ level 7 |
| | | 8 | S | Enable bit set for DMA engine |
| | | 9 | S | Enable bit set for general purpose input 0 |
| | | 10 | S | Enable bit set for general purpose input 1 |
| | | 11 | S | General purpose output bit 0 |
| | | 12 | S | General purpose output bit 1 |
| 0001220 | INT_ENABLECLR | 0 | C | Enable bit clear for VMECC errors |
| | | 1 | C | Enable bit clear for VME IRQ level 1 |
| | | 2 | C | Enable bit clear for VME IRQ level 2 |
| | | 3 | C | Enable bit clear for VME IRQ level 3 |
| | | 4 | C | Enable bit clear for VME IRQ level 4 |
| | | 5 | C | Enable bit clear for VME IRQ level 5 |
| | | 6 | C | Enable bit clear for VME IRQ level 6 |
| | | 7 | C | Enable bit clear for VME IRQ level 7 |
| | | 8 | C | Enable bit clear for DMA engine |
| | | 9 | C | Enable bit clear for general purpose input 0 |
| | | 10 | C | Enable bit clear for general purpose input 1 |
| | | 11 | C | General purpose output bit 0 |
| | | 12 | C | General purpose output bit 1 |

### 2.5.6 PIO Mapping and PIO Timer

This group is distinguished from the configuration group by the possibility that it might be written after initialization time. As described in the PIO Mapping section, there are 15 locations in a PIO mapping RAM. Each location contains six AM bits and nine address bits to extend the address to 32 bits. The PIO mapping allows access to 15 times 8Mb of VME address space at one time. Usually, this will be enough even if the mapping is allocated statically. However, if more address space is needed, software could, with locks, allow one or more of the map locations to be changed during system operation.

Also in this group is the PIO delay register. Properly designed VME boards should not require this register since a board should factor any required delays between accesses into its speed of handshakes. However, boards exist that require a certain amount of delay between accesses. Because of the multiple buses and fifos between the R4000s and the VME bus, it is impossible to program a delay directly from the R4000s, so this register is needed. A write with a given value to this location causes delay of that many ticks before another PIO operation of any type can proceed. The unit of time is 16 of the VMECC's clock periods; if the VMECC is running at 50MHz, this is 320ns. The value written is an 8 bit value, so this allows delays of at least 80 microseconds.

**Table 8: PIO Mapping and Timer Group**

| Address | Register Name | Bits | R/W | Function |
|---|---|---|---|---|
| 0001180 | PIOTIMER | 7:0 | W | Delay all PIOs for number of ticks in data field |
| 0001380 | PIOMAP0 | 15:0 | R/W | initialize to all zeros; used for A16 upper address bits on bus (really shouldn't be used by A16 slaves but zero it anyway) |
| 0001388 | PIOMAP1 | 8:0 | R/W | extend Big Window PIO addresses to 32 bits with these bits in A31-A23 for 00800000-00FFFFFF |
| | | 9 | R/W | reserved |
| | | 15:10 | R/W | AM bits for this access |
| 0001390 | PIOMAP2 | 15:0[1] | R/W | PIO mapping bits for 01000000 - 017FFFFF |
| 0001398 | PIOMAP3 | 15:0 | R/W | PIO mapping bits for 01800000 - 01FFFFFF |
| 00013A0 | PIOMAP4 | 15:0 | R/W | PIO mapping bits for 02000000 - 027FFFFF |
| 00013A8 | PIOMAP5 | 15:0 | R/W | PIO mapping bits for 02800000 - 02FFFFFF |
| 00013B0 | PIOMAP6 | 15:0 | R/W | PIO mapping bits for 03000000 - 037FFFFF |
| 00013B8 | PIOMAP7 | 15:0 | R/W | PIO mapping bits for 03800000 - 03FFFFFF |
| 00013C0 | PIOMAP8 | 15:0 | R/W | PIO mapping bits for 04000000 - 047FFFFF |
| 00013C8 | PIOMAP9 | 15:0 | R/W | PIO mapping bits for 04800000 - 04FFFFFF |
| 00013D0 | PIOMAP10 | 15:0 | R/W | PIO mapping bits for 05000000 - 057FFFFF |
| 00013D8 | PIOMAP11 | 15:0 | R/W | PIO mapping bits for 05800000 - 05FFFFFF |
| 00013E0 | PIOMAP12 | 15:0 | R/W | PIO mapping bits for 06000000 - 067FFFFF |
| 00013E8 | PIOMAP13 | 15:0 | R/W | PIO mapping bits for 06800000 - 06FFFFFF |
| 00013F0 | PIOMAP14 | 15:0 | R/W | PIO mapping bits for 07000000 - 077FFFFF |
| 00013F8 | PIOMAP15 | 15:0 | R/W | PIO mapping bits for 07800000 - 07FFFFFF |

1. PIOMAP2 - PIOMAP15 have the same bit assignments detailed for PIOMAP1

## 2.6 Diagnostics Features

By setting the PIO map and slave enable bits appropriately, PIO loop-backs going out to the VME bus itself are possible in both directions involving PIOs and DMA. The interrupt request bit register is writable so that EBUS interrupt vectors can be artificially scheduled to test the interrupt vector values and mechanism. The DMA engine cannot be used for loop-backs.

## 2.7 Error Handling

The VMECC has a register to capture FCI error conditons and VME buserror occurence. It also has two registers to capture the VME address, AM bits, DS0/1, LW, etc. These registers capture the first errors on each side until rearmed.

The basic strategy for errors is that errors that are bad FCI data parity, or errors originating on the VME bus (timeouts) cause an interrupt, while FCI command parity errors or FCI protocol timeouts (no handshake or data after a request) cause a request to the F-chip to reset the FCI bus. The latter operation in turn causes the F-chip to cause an interrupt. DMA read data with bad parity is passed through to the VME bus with the VME Bus Error asserted to inform the controller; it presumably will interrupt the system. PIO operations with bad data parity on address or data will not be put onto the VME bus; they will cause an interrupt and any such bad PIO Reads will be allowed to time out way back at the R4000.

## 2.8 Bi-Endian Support

The DMA input and output fifos have byte swappers between them and the VME bus. Finding and delivering data within the 64 bit FCI double words for PIOs is endian sensitive. The address of a VME PIO operation is copied to the VME bus; where the data is found or delivered on the FCI bus will depend on the endian mode. Basically, DMA data is byte-swapped and PIO operations are not logically swapped.

## 2.9 Device ID

The VMECC returns an eight bit device ID in response to a FCI reset command. This ID is divided into four bit major and minor fields.

**Table 9: Device ID**

| Major ID (Bits 7:4) | Minor ID (Bits 3:0) | Description |
|---|---|---|
| 0x1 | 0x1 | VMECC with VME bus attached |
| | 0x2 | VMECC with HIPPI attached |

# CHAPTER 3 Hardware Specification

## 3.1 Block diagram

The VMECC block diagram shows the major sections of the chip and major interconnects. The block title “local registers” actually includes many of the other blocks such as the A64 Upper Address register, etc.

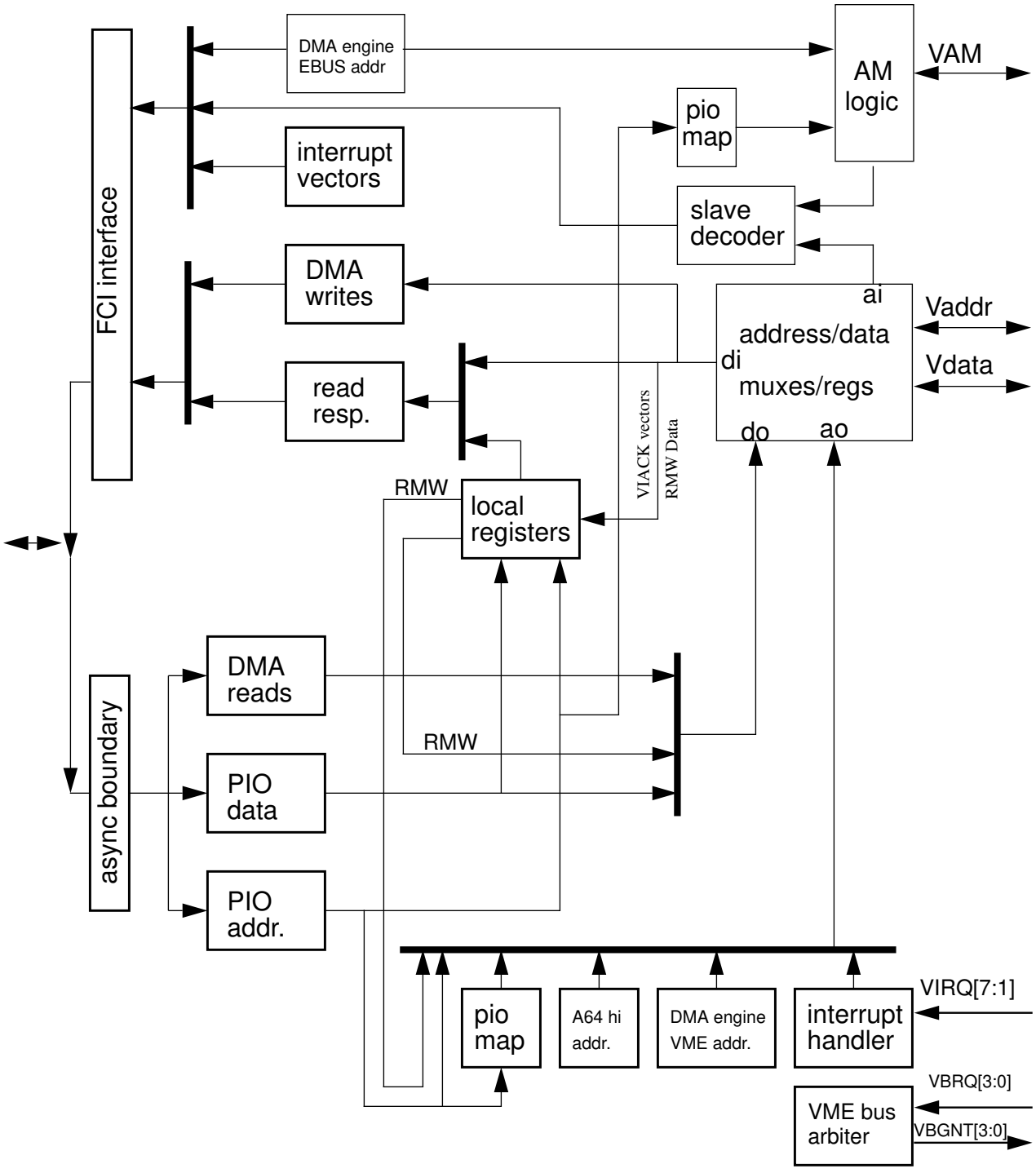


**Figure 2** VMECC Block Diagram

### 3.1.1 FCI Input Block

#### 3.1.1.1 Input Synchronizer

#### 3.1.1.2 PIO Address Fifo

#### 3.1.1.3 PIO Data Fifo

### 3.1.2 FCI Output Block

#### 3.1.2.1 Interrupt Vectors

#### 3.1.2.2 DMA Write Fifo

#### 3.1.2.3 Read Responses

### 3.1.3 Local Registers

### 3.1.4 VME Bus Multiplexers and Registers

### 3.1.5 Slave Decoder

### 3.1.6 VME Arbiter

### 3.1.7 Interrupt Handler

## 3.2 FCI Interface

The FCI interface protocol is described in the FCI Specification.

## 3.3 VME Interface

The VME protocol is described in the VMEbus Specification C.1 and D.

# CHAPTER 4 Testability

## 4.1 Diagnostics

The VMECC supports VME PIO loopbacks, where a PIO operation can cause a DMA operation to system memory. RMW operation can also do loopback tests. The INT_REQUESTSM register greatly aids in checking the interrupt system and the LOOPBACKTRIG location helps to exercise the FCI AuxCmd bus more easily.

## 4.2 Scan

The VMECC has JTAG boundary scan and (almost) all internal register bits are scannable.

# CHAPTER 5 Implementation

## 5.1 Technology

LSI Logic 200K technology, 182K gate die.

## 5.2 Gate Count

77K used gates

## 5.3 Package Type

299 PPGA

## 5.4 Pin List

The following three tables summarize the pins of the VMECC.

### 5.4.1 FCI Side

**Table 10: FCI Side Pins**

| Pins | Name | Used For |
|---|---|---|
| 32 | FCIDB_L | data |
| 2 | FCIDBP_L | parity on data pins FCIDB |
| 1 | FCIMCLK_H | master clock; sample clock for data leaving VMECC |
| 1 | FCIMCLK_L | master clock; sample clock for data leaving VMECC |
| 1 | FCIMOKTD | signal from VMECC to F-chip that it's "OK to drive" data bus |
| 2 | FCIMCMD_L | command bits from VMECC |
| 8 | FCIMAUXCMD_L | byte selects, read resp. #, interrupt or address from VMECC |
| 1 | FCIMCMDP_L | parity bit on master (VMECC) all command pins |
| 1 | FCIMVREF | NTL voltage reference for the VMECC outputs |
| 1 | FCISCLK_H | slave clock; sample clock for data leaving F-chip |
| 1 | FCISCLK_L | slave clock; sample clock for data leaving F-chip |
| 1 | FCISOE | signal that F-chip is driving FCIDB data bus |
| 2 | FCISCMD_L | command bits from F-chip |
| 1 | FCISCMDP_L | parity bit on slave (F-chip) all command pins |
| 1 | FCISVREF | NTL voltage reference for F chip outputs |
| 1 | FCISBEND_L | when true, system is big-endian |
| 1 | FCIMDOE_L | auxiliary wire to control actual external drivers |
| 58 | Total Pins | |

### 5.4.2 VME Side

**Table 11: VME Side Pins**

| Bits | Signal Name | Used For |
|---|---|---|
| 32 | VDATA[31:0] | data bits D31-D0 and A64 address bits A63-A32 |
| 1 | VDATA_OUT_EN | enable for F623 to drive from VMECC to VMEbus |
| 1 | VDATA_IN_EN | enable for F623 to drive from VMEbus to VMECC |
| 31 | VADDR[31:1] | address bits A31-A1 and D64 data bits D63-D33 |
| 1 | VLWORD | long word and D64 data bit D32 (part of vaddr group) |
| 1 | VADDR_OUT_EN | enable for F623 to drive from VMECC to VMEbus |
| 1 | VADDR_IN_EN | enable for F623 to drive from VMEbus to VMECC |
| 1 | VWRITE_OUT_L | direction of transfer driven when VMECC is master |
| 1 | VDS0_OUT_L | lower data strobe driven when VMECC is master |
| 1 | VDS1_OUT_L | upper data strobe driven when VMECC is master |
| 1 | VAS_OUT_L | address strobe driven when VMECC is master |
| 6 | VAM[5:0] | address modifier bits |
| 1 | VIACK_L | the bussed IACK signal that invalidates A31-A4 and AM5-0; driven onto bus by same F623 part as VAM |
| 1 | VAM_OUT_EN_L | enable for F623 to drive from VMECC to VMEbus (also output enable for F244 to drive AS, DS1, DS0, WRITE) |
| 1 | VAM_IN_EN | enable for F623 to drive from VMEbus to VMECC |
| 1 | VWRITE_IN | direction of transfer on VMEbus |
| 1 | VDS0_IN | lower data strobe |
| 1 | VDS1_IN | upper data strobe |
| 1 | VDSX | the OR of DS1 and DS0 |
| 1 | VDSX_D20 | VDSX delayed 20ns |
| 1 | VAS_IN | address strobe |
| 1 | VBERR_IN | VME bus error input |

**Table 11: VME Side Pins (Continued)**

| Bits | Signal Name | Used For |
|---|---|---|
| 1 | VBBSY_IN | VME bus busy input |
| 1 | VDTACK_IN | data acknowledge |
| 1 | VBERR_OUT | O.C. VME bus error when VMECC is slave or by bus timer |
| 1 | VBBSY_OUT | O.C. VME bus busy output when VMECC is master |
| 1 | VMYREQ | O.C. VBR3&2 driven when an internal master wants service |
| 1 | VCLR_OUT | VME bus clear driven by internal arbiter |
| 1 | VME_RESET_OUT_L | reset out |
| 1 | VME_RESET_IN_L | reset in; probably from a PON low voltage detector chip |
| 7 | VIRQ[7:1] | VME interrupt request line inputs |
| 4 | VBRQ_IN[3:0] | VME bus request line inputs |
| 4 | VGNT_OUT[3:0] | VME bus grant outputs from internal arbiter |
| 2 | AUX_INTR[1:0] | auxiliary interrupt pins |
| 2 | AUX_OUT[1:0] | auxiliary output pins |
| 1 | WT_CLK_A | clock for A-side input registers |
| 1 | WT_CLK_B | clock for B-side input registers |
| 1 | WT_CLK_A_LOOP | WT_CLK_A looped through chip to tell the external handshake logic that the chip has seen it |
| 1 | WT_CLK_B_LOOP | WT_CLK_B looped through chip to tell the external handshake logic that the chip has seen it |
| 1 | SLAVEON | allows external slave logic to handshake; signal that VMECC has recognized and prepared for a slave response |
| 1 | SLAVE_WRITE | the direction the VMECC is expecting for a slave response; helps to detect RMW slave cycles |
| 1 | CURRENT_CYCLE | output of external asynchronous state machine -- this is the same VME cycle that the VMECC has recognized |
| 1 | D64_SLAVE_READ | indicates that the VMEA buffers should drive the VME address bus as well as the data bus during slave reads |

**Table 11: VME Side Pins (Continued)**

| Bits | Signal Name | Used For |
|---|---|---|
| 1 | FLAG_A | one bit of the two bit grey counter; how far the external asynchronous logic has progressed |
| 1 | FLAG_B | one bit of the two bit grey counter; how far the external asynchronous logic has progressed |
| 1 | CLR_GREY_CNTR | clears the external two bit grey counter FLAG_A/FLAG_B |
| 1 | ACK_A | one of two signals to external handshake logic telling it how far it may proceed |
| 1 | ACK_B | one of two signals to external handshake logic telling it how far it may proceed |
| 1 | DTACK_OUT | allows a handshake to acknowledge first A64/D64 cycle |
| 1 | HIPPI_MODE_L | shortens handshakes and alters prefetch strategy |
| 1 | VCLOCK_OUT | internal clock brought out for synchrfonous interface (HIPPI board) |
| 131 | Total Pins | |

### 5.4.3 Pin Count Summary

The VMECC fits in a 299 pin PGA.

**Table 12: Total Pins**

| Pins | Name | Used For |
|---|---|---|
| 1 | clock_in | clock input |
| 7 | PLL | PLL control pins for FCI input clock |
| 6 | SCAN / NAND | JTAG pins and LSI NAND-tree |
| 29 | FCI-BUS/2 | grounds for NTL pins |
| 4 | VSS-VME | grounds for TTL side pins |
| 8 | VCC2 | 5V for TTL side pins |
| 6 | VSS | core grounds |
| 6 | VCC | core VCC |
| 58 | FCI-side | FCI bus signals |
| 131 | VME-side | VME pins |
| 273 | Total Pins | |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| ACK_A | C | 10 | OUT |
| ACK_B | A | 8 | OUT |
| AUX_INTR_0_ | A | 15 | IN |
| AUX_INTR_1_ | E | 12 | IN |
| AUX_OUT_0_ | B | 14 | OUT |
| AUX_OUT_1_ | D | 12 | OUT |
| CLOCK2X | U | 12 | IN |
| CLOCK_RESET_L | T | 12 | IN |
| CLR_GREY_CNTR | C | 11 | OUT |
| CURRENT_CYCLE | V | 12 | IN |
| D64_SLAVE_READ | B | 13 | OUT |
| DTACK_OUT | E | 11 | OUT |
| FCIDBP_L_0_ | J | 17 | BIDIR |
| FCIDBP_L_1_ | U | 15 | BIDIR |
| FCIDB_L_0_ | B | 15 | BIDIR |
| FCIDB_L_10_ | D | 19 | BIDIR |
| FCIDB_L_11_ | E | 18 | BIDIR |
| FCIDB_L_12_ | D | 20 | BIDIR |
| FCIDB_L_13_ | E | 19 | BIDIR |
| FCIDB_L_14_ | F | 18 | BIDIR |
| FCIDB_L_15_ | E | 20 | BIDIR |
| FCIDB_L_16_ | P | 18 | BIDIR |
| FCIDB_L_17_ | R | 19 | BIDIR |
| FCIDB_L_18_ | J | 18 | BIDIR |
| FCIDB_L_19_ | R | 18 | BIDIR |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| FCIDB_L_1_ | A | 16 | BIDIR |
| FCIDB_L_20_ | K | 16 | BIDIR |
| FCIDB_L_21_ | K | 18 | BIDIR |
| FCIDB_L_22_ | D | 16 | BIDIR |
| FCIDB_L_23_ | U | 19 | BIDIR |
| FCIDB_L_24_ | V | 17 | BIDIR |
| FCIDB_L_25_ | U | 16 | BIDIR |
| FCIDB_L_26_ | W | 17 | BIDIR |
| FCIDB_L_27_ | V | 16 | BIDIR |
| FCIDB_L_28_ | W | 16 | BIDIR |
| FCIDB_L_29_ | X | 16 | BIDIR |
| FCIDB_L_2_ | C | 15 | BIDIR |
| FCIDB_L_30_ | V | 15 | BIDIR |
| FCIDB_L_31_ | W | 15 | BIDIR |
| FCIDB_L_3_ | B | 16 | BIDIR |
| FCIDB_L_4_ | A | 17 | BIDIR |
| FCIDB_L_5_ | C | 16 | BIDIR |
| FCIDB_L_6_ | B | 17 | BIDIR |
| FCIDB_L_7_ | T | 18 | BIDIR |
| FCIDB_L_8_ | D | 18 | BIDIR |
| FCIDB_L_9_ | E | 17 | BIDIR |
| FCIMAUXCMD_L_0_ | T | 17 | OUT |
| FCIMAUXCMD_L_1_ | U | 18 | OUT |
| FCIMAUXCMD_L_2_ | V | 19 | OUT |
| FCIMAUXCMD_L_3_ | U | 17 | OUT |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| FCIMAUXCMD_L_4_ | W | 19 | OUT |
| FCIMAUXCMD_L_5_ | T | 20 | OUT |
| FCIMAUXCMD_L_6_ | U | 20 | OUT |
| FCIMAUXCMD_L_7_ | T | 19 | OUT |
| FCIMCLK_H | K | 17 | OUT |
| FCIMCLK_L | K | 19 | OUT |
| FCIMCMDP_L | M | 17 | OUT |
| FCIMCMD_L_0_ | M | 16 | OUT |
| FCIMCMD_L_1_ | N | 17 | OUT |
| FCIMDOE_L | N | 16 | OUT |
| FCIMOKTD | M | 18 | OUT |
| FCISBEND_L | L | 19 | IN |
| FCISCLK_H | E | 16 | IN |
| FCISCLK_L | F | 16 | IN |
| FCISCMDP_L | L | 16 | IN |
| FCISCMD_L_0_ | M | 19 | IN |
| FCISCMD_L_1_ | L | 18 | IN |
| FCISOE | N | 20 | IN |
| FCISVREF | L | 17 | IN |
| FLAG_A | C | 12 | IN |
| FLAG_B | A | 14 | IN |
| HIPPI_MODE_L | C | 13 | IN |
| IDDTEST_L | G | 17 | IN |
| JTAG_TCK | H | 16 | IN |
| JTAG_TDI | G | 18 | IN |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| JTAG_TDO | F | 20 | OUT |
| JTAG_TMS | H | 17 | IN |
| LP1 | E | 15 | OUT |
| LP2 | E | 14 | IN |
| PLLBYPASS_L | C | 18 | IN |
| PLLTESTOUT | C | 17 | OUT |
| PLLVDD | B | 19 | IN |
| PLLVSS | D | 15 | IN |
| PLLVSSAGND | B | 18 | OUT |
| RAMTEST_L | U | 13 | IN |
| SCLK2X_OUT | T | 13 | IN |
| SCLR_L | V | 14 | IN |
| SLAVEON | B | 12 | OUT |
| SLAVE_WRITE | D | 11 | OUT |
| TESTOUT | J | 16 | OUT |
| TEST_L | V | 10 | IN |
| VADDR_10_ | K | 3 | BIDIR |
| VADDR_11_ | J | 3 | BIDIR |
| VADDR_12_ | J | 5 | BIDIR |
| VADDR_13_ | J | 2 | BIDIR |
| VADDR_14_ | K | 2 | BIDIR |
| VADDR_15_ | L | 2 | BIDIR |
| VADDR_16_ | M | 2 | BIDIR |
| VADDR_17_ | N | 2 | BIDIR |
| VADDR_18_ | P | 1 | BIDIR |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| VADDR_19_ | N | 3 | BIDIR |
| VADDR_1_ | L | 5 | BIDIR |
| VADDR_20_ | P | 3 | BIDIR |
| VADDR_21_ | R | 2 | BIDIR |
| VADDR_22_ | P | 4 | BIDIR |
| VADDR_23_ | T | 1 | BIDIR |
| VADDR_24_ | L | 4 | BIDIR |
| VADDR_25_ | H | 3 | BIDIR |
| VADDR_26_ | H | 4 | BIDIR |
| VADDR_27_ | H | 5 | BIDIR |
| VADDR_28_ | H | 1 | BIDIR |
| VADDR_29_ | G | 1 | BIDIR |
| VADDR_2_ | L | 3 | BIDIR |
| VADDR_30_ | G | 2 | BIDIR |
| VADDR_31_ | G | 3 | BIDIR |
| VADDR_3_ | M | 3 | BIDIR |
| VADDR_4_ | M | 5 | BIDIR |
| VADDR_5_ | R | 1 | BIDIR |
| VADDR_6_ | N | 4 | BIDIR |
| VADDR_7_ | N | 5 | BIDIR |
| VADDR_8_ | K | 4 | BIDIR |
| VADDR_9_ | K | 5 | BIDIR |
| VADDR_IN_EN | C | 8 | OUT |
| VADDR_OUT_EN | E | 9 | OUT |
| VAM_0_ | W | 3 | BIDIR |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| VAM_1_ | V | 4 | BIDIR |
| VAM_2_ | U | 5 | BIDIR |
| VAM_3_ | W | 4 | BIDIR |
| VAM_4_ | X | 4 | BIDIR |
| VAM_5_ | W | 5 | BIDIR |
| VAM_IN_EN | W | 11 | OUT |
| VAM_OUT_EN_L | U | 10 | OUT |
| VAS_IN | C | 9 | IN |
| VAS_OUT_L | A | 7 | OUT |
| VBBSY_IN | E | 8 | IN |
| VBBSY_OUT | D | 8 | OUT |
| VBERR_IN | C | 7 | IN |
| VBERR_OUT | B | 7 | OUT |
| VBRQ_IN_0_ | T | 8 | IN |
| VBRQ_IN_1_ | U | 8 | IN |
| VBRQ_IN_2_ | V | 8 | IN |
| VBRQ_IN_3_ | T | 9 | IN |
| VCLOCK_OUT | A | 6 | OUT |
| VCLR_OUT | D | 7 | OUT |
| VDATA_0_ | U | 7 | BIDIR |
| VDATA_10_ | U | 3 | BIDIR |
| VDATA_11_ | T | 4 | BIDIR |
| VDATA_12_ | T | 3 | BIDIR |
| VDATA_13_ | U | 1 | BIDIR |
| VDATA_14_ | T | 2 | BIDIR |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| VDATA_15_ | R | 3 | BIDIR |
| VDATA_16_ | G | 4 | BIDIR |
| VDATA_17_ | G | 5 | BIDIR |
| VDATA_18_ | F | 4 | BIDIR |
| VDATA_19_ | E | 4 | BIDIR |
| VDATA_1_ | T | 7 | BIDIR |
| VDATA_20_ | E | 5 | BIDIR |
| VDATA_21_ | E | 6 | BIDIR |
| VDATA_22_ | D | 6 | BIDIR |
| VDATA_23_ | E | 7 | BIDIR |
| VDATA_24_ | F | 1 | BIDIR |
| VDATA_25_ | F | 2 | BIDIR |
| VDATA_26_ | F | 3 | BIDIR |
| VDATA_27_ | E | 1 | BIDIR |
| VDATA_28_ | E | 3 | BIDIR |
| VDATA_29_ | D | 2 | BIDIR |
| VDATA_2_ | U | 6 | BIDIR |
| VDATA_30_ | D | 3 | BIDIR |
| VDATA_31_ | C | 2 | BIDIR |
| VDATA_3_ | T | 6 | BIDIR |
| VDATA_4_ | T | 5 | BIDIR |
| VDATA_5_ | R | 5 | BIDIR |
| VDATA_6_ | R | 4 | BIDIR |
| VDATA_7_ | P | 5 | BIDIR |
| VDATA_8_ | W | 2 | BIDIR |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| VDATA_9_ | V | 2 | BIDIR |
| VDATA_IN_EN | B | 6 | OUT |
| VDATA_OUT_EN | C | 6 | OUT |
| VDD | B | 4 | IN |
| VDD | B | 8 | IN |
| VDD | E | 2 | IN |
| VDD | H | 2 | IN |
| VDD | M | 4 | IN |
| VDD | U | 2 | IN |
| VDD | U | 9 | IN |
| VDD | V | 5 | IN |
| VDS0_IN | W | 8 | IN |
| VDS0_OUT_L | A | 5 | OUT |
| VDS1_IN | V | 9 | IN |
| VDS1_OUT_L | B | 5 | OUT |
| VDSX | W | 9 | IN |
| VDSX_D20 | W | 10 | IN |
| VDTACK_IN | B | 11 | IN |
| VGNT_OUT_0_ | W | 6 | OUT |
| VGNT_OUT_1_ | V | 7 | OUT |
| VGNT_OUT_2_ | X | 6 | OUT |
| VGNT_OUT_3_ | W | 7 | OUT |
| VIACK_L | X | 5 | BIDIR |
| VIRQ_1_ | W | 12 | IN |
| VIRQ_2_ | U | 11 | IN |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| VIRQ_3_ | X | 13 | IN |
| VIRQ_4_ | T | 11 | IN |
| VIRQ_5_ | W | 13 | IN |
| VIRQ_6_ | V | 11 | IN |
| VIRQ_7_ | V | 13 | IN |
| VLWORD | V | 6 | BIDIR |
| VME_RESET_IN_L | C | 4 | IN |
| VME_RESET_OUT_L | C | 5 | OUT |
| VMYREQ | D | 5 | OUT |
| VSS | C | 3 | IN |
| VSS | C | 19 | IN |
| VSS | D | 9 | IN |
| VSS | D | 14 | IN |
| VSS | E | 13 | IN |
| VSS | F | 5 | IN |
| VSS | F | 17 | IN |
| VSS | F | 19 | IN |
| VSS | G | 16 | IN |
| VSS | G | 19 | IN |
| VSS | H | 18 | IN |
| VSS | H | 19 | IN |
| VSS | J | 4 | IN |
| VSS | J | 19 | IN |
| VSS | N | 18 | IN |
| VSS | N | 19 | IN |

**Table 13: Chip Pinout by Name**

| name | pin | | direction |
|---|---|---|---|
| VSS | P | 16 | IN |
| VSS | P | 17 | IN |
| VSS | P | 20 | IN |
| VSS | R | 16 | IN |
| VSS | R | 17 | IN |
| VSS | T | 14 | IN |
| VSS | T | 16 | IN |
| VSS | U | 14 | IN |
| VSS | V | 3 | IN |
| VSS | W | 14 | IN |
| VSS | W | 18 | IN |
| VSS | X | 7 | IN |
| VSS | X | 14 | IN |
| VSS2 | G | 20 | IN |
| VSS2 | P | 2 | IN |
| VSS2 | P | 19 | IN |
| VSS2 | R | 20 | IN |
| VWRITE_IN | T | 10 | IN |
| VWRITE_OUT_L | B | 3 | OUT |
| WT_CLK_A | E | 10 | IN |
| WT_CLK_A_LOOP | D | 10 | OUT |
| WT_CLK_B | B | 9 | IN |
| WT_CLK_B_LOOP | B | 10 | OUT |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| VDS0_OUT_L | A | 5 | OUT |
| VCLOCK_OUT | A | 6 | OUT |
| VAS_OUT_L | A | 7 | OUT |
| ACK_B | A | 8 | OUT |
| FLAG_B | A | 14 | IN |
| AUX_INTR_0_ | A | 15 | IN |
| FCIDB_L_1_ | A | 16 | BIDIR |
| FCIDB_L_4_ | A | 17 | BIDIR |
| VWRITE_OUT_L | B | 3 | OUT |
| VDD | B | 4 | IN |
| VDS1_OUT_L | B | 5 | OUT |
| VDATA_IN_EN | B | 6 | OUT |
| VBERR_OUT | B | 7 | OUT |
| VDD | B | 8 | IN |
| WT_CLK_B | B | 9 | IN |
| WT_CLK_B_LOOP | B | 10 | OUT |
| VDTACK_IN | B | 11 | IN |
| SLAVEON | B | 12 | OUT |
| D64_SLAVE_READ | B | 13 | OUT |
| AUX_OUT_0_ | B | 14 | OUT |
| FCIDB_L_0_ | B | 15 | BIDIR |
| FCIDB_L_3_ | B | 16 | BIDIR |
| FCIDB_L_6_ | B | 17 | BIDIR |
| PLLVSSAGND | B | 18 | OUT |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| PLLVDD | B | 19 | IN |
| VDATA_31_ | C | 2 | BIDIR |
| VSS | C | 3 | IN |
| VME_RESET_IN_L | C | 4 | IN |
| VME_RESET_OUT_L | C | 5 | OUT |
| VDATA_OUT_EN | C | 6 | OUT |
| VBERR_IN | C | 7 | IN |
| VADDR_IN_EN | C | 8 | OUT |
| VAS_IN | C | 9 | IN |
| ACK_A | C | 10 | OUT |
| CLR_GREY_CNTR | C | 11 | OUT |
| FLAG_A | C | 12 | IN |
| HIPPI_MODE_L | C | 13 | IN |
| FCIDB_L_2_ | C | 15 | BIDIR |
| FCIDB_L_5_ | C | 16 | BIDIR |
| PLLTESTOUT | C | 17 | OUT |
| PLLBYPASS_L | C | 18 | IN |
| VSS | C | 19 | IN |
| VDATA_29_ | D | 2 | BIDIR |
| VDATA_30_ | D | 3 | BIDIR |
| VMYREQ | D | 5 | OUT |
| VDATA_22_ | D | 6 | BIDIR |
| VCLR_OUT | D | 7 | OUT |
| VBBSY_OUT | D | 8 | OUT |
| VSS | D | 9 | IN |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| WT_CLK_A_LOOP | D | 10 | OUT |
| SLAVE_WRITE | D | 11 | OUT |
| AUX_OUT_1_ | D | 12 | OUT |
| VSS | D | 14 | IN |
| PLLVSS | D | 15 | IN |
| FCIDB_L_22_ | D | 16 | BIDIR |
| FCIDB_L_8_ | D | 18 | BIDIR |
| FCIDB_L_10_ | D | 19 | BIDIR |
| FCIDB_L_12_ | D | 20 | BIDIR |
| VDATA_27_ | E | 1 | BIDIR |
| VDD | E | 2 | IN |
| VDATA_28_ | E | 3 | BIDIR |
| VDATA_19_ | E | 4 | BIDIR |
| VDATA_20_ | E | 5 | BIDIR |
| VDATA_21_ | E | 6 | BIDIR |
| VDATA_23_ | E | 7 | BIDIR |
| VBBSY_IN | E | 8 | IN |
| VADDR_OUT_EN | E | 9 | OUT |
| WT_CLK_A | E | 10 | IN |
| DTACK_OUT | E | 11 | OUT |
| AUX_INTR_1_ | E | 12 | IN |
| VSS | E | 13 | IN |
| LP2 | E | 14 | IN |
| LP1 | E | 15 | OUT |
| FCISCLK_H | E | 16 | IN |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| FCIDB_L_9_ | E | 17 | BIDIR |
| FCIDB_L_11_ | E | 18 | BIDIR |
| FCIDB_L_13_ | E | 19 | BIDIR |
| FCIDB_L_15_ | E | 20 | BIDIR |
| VDATA_24_ | F | 1 | BIDIR |
| VDATA_25_ | F | 2 | BIDIR |
| VDATA_26_ | F | 3 | BIDIR |
| VDATA_18_ | F | 4 | BIDIR |
| VSS | F | 5 | IN |
| FCISCLK_L | F | 16 | IN |
| VSS | F | 17 | IN |
| FCIDB_L_14_ | F | 18 | BIDIR |
| VSS | F | 19 | IN |
| JTAG_TDO | F | 20 | OUT |
| VADDR_29_ | G | 1 | BIDIR |
| VADDR_30_ | G | 2 | BIDIR |
| VADDR_31_ | G | 3 | BIDIR |
| VDATA_16_ | G | 4 | BIDIR |
| VDATA_17_ | G | 5 | BIDIR |
| VSS | G | 16 | IN |
| IDDTEST_L | G | 17 | IN |
| JTAG_TDI | G | 18 | IN |
| VSS | G | 19 | IN |
| VSS2 | G | 20 | IN |
| VADDR_28_ | H | 1 | BIDIR |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| VDD | H | 2 | IN |
| VADDR_25_ | H | 3 | BIDIR |
| VADDR_26_ | H | 4 | BIDIR |
| VADDR_27_ | H | 5 | BIDIR |
| JTAG_TCK | H | 16 | IN |
| JTAG_TMS | H | 17 | IN |
| VSS | H | 18 | IN |
| VSS | H | 19 | IN |
| VADDR_13_ | J | 2 | BIDIR |
| VADDR_11_ | J | 3 | BIDIR |
| VSS | J | 4 | IN |
| VADDR_12_ | J | 5 | BIDIR |
| TESTOUT | J | 16 | OUT |
| FCIDBP_L_0_ | J | 17 | BIDIR |
| FCIDB_L_18_ | J | 18 | BIDIR |
| VSS | J | 19 | IN |
| VADDR_14_ | K | 2 | BIDIR |
| VADDR_10_ | K | 3 | BIDIR |
| VADDR_8_ | K | 4 | BIDIR |
| VADDR_9_ | K | 5 | BIDIR |
| FCIDB_L_20_ | K | 16 | BIDIR |
| FCIMCLK_H | K | 17 | OUT |
| FCIDB_L_21_ | K | 18 | BIDIR |
| FCIMCLK_L | K | 19 | OUT |
| VADDR_15_ | L | 2 | BIDIR |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| VADDR_2_ | L | 3 | BIDIR |
| VADDR_24_ | L | 4 | BIDIR |
| VADDR_1_ | L | 5 | BIDIR |
| FCISCMDP_L | L | 16 | IN |
| FCISVREF | L | 17 | IN |
| FCISCMD_L_1_ | L | 18 | IN |
| FCISBEND_L | L | 19 | IN |
| VADDR_16_ | M | 2 | BIDIR |
| VADDR_3_ | M | 3 | BIDIR |
| VDD | M | 4 | IN |
| VADDR_4_ | M | 5 | BIDIR |
| FCIMCMD_L_0_ | M | 16 | OUT |
| FCIMCMDP_L | M | 17 | OUT |
| FCIMOKTD | M | 18 | OUT |
| FCISCMD_L_0_ | M | 19 | IN |
| VADDR_17_ | N | 2 | BIDIR |
| VADDR_19_ | N | 3 | BIDIR |
| VADDR_6_ | N | 4 | BIDIR |
| VADDR_7_ | N | 5 | BIDIR |
| FCIMDOE_L | N | 16 | OUT |
| FCIMCMD_L_1_ | N | 17 | OUT |
| VSS | N | 18 | IN |
| VSS | N | 19 | IN |
| FCISOE | N | 20 | IN |
| VADDR_18_ | P | 1 | BIDIR |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| VSS2 | P | 2 | IN |
| VADDR_20_ | P | 3 | BIDIR |
| VADDR_22_ | P | 4 | BIDIR |
| VDATA_7_ | P | 5 | BIDIR |
| VSS | P | 16 | IN |
| VSS | P | 17 | IN |
| FCIDB_L_16_ | P | 18 | BIDIR |
| VSS2 | P | 19 | IN |
| VSS | P | 20 | IN |
| VADDR_5_ | R | 1 | BIDIR |
| VADDR_21_ | R | 2 | BIDIR |
| VDATA_15_ | R | 3 | BIDIR |
| VDATA_6_ | R | 4 | BIDIR |
| VDATA_5_ | R | 5 | BIDIR |
| VSS | R | 16 | IN |
| VSS | R | 17 | IN |
| FCIDB_L_19_ | R | 18 | BIDIR |
| FCIDB_L_17_ | R | 19 | BIDIR |
| VSS2 | R | 20 | IN |
| VADDR_23_ | T | 1 | BIDIR |
| VDATA_14_ | T | 2 | BIDIR |
| VDATA_12_ | T | 3 | BIDIR |
| VDATA_11_ | T | 4 | BIDIR |
| VDATA_4_ | T | 5 | BIDIR |
| VDATA_3_ | T | 6 | BIDIR |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| VDATA_1_ | T | 7 | BIDIR |
| VBRQ_IN_0_ | T | 8 | IN |
| VBRQ_IN_3_ | T | 9 | IN |
| VWRITE_IN | T | 10 | IN |
| VIRQ_4_ | T | 11 | IN |
| CLOCK_RESET_L | T | 12 | IN |
| SCLK2X_OUT | T | 13 | IN |
| VSS | T | 14 | IN |
| VSS | T | 16 | IN |
| FCIMAUXCMD_L_0_ | T | 17 | OUT |
| FCIDB_L_7_ | T | 18 | BIDIR |
| FCIMAUXCMD_L_7_ | T | 19 | OUT |
| FCIMAUXCMD_L_5_ | T | 20 | OUT |
| VDATA_13_ | U | 1 | BIDIR |
| VDD | U | 2 | IN |
| VDATA_10_ | U | 3 | BIDIR |
| VAM_2_ | U | 5 | BIDIR |
| VDATA_2_ | U | 6 | BIDIR |
| VDATA_0_ | U | 7 | BIDIR |
| VBRQ_IN_1_ | U | 8 | IN |
| VDD | U | 9 | IN |
| VAM_OUT_EN_L | U | 10 | OUT |
| VIRQ_2_ | U | 11 | IN |
| CLOCK2X | U | 12 | IN |
| RAMTEST_L | U | 13 | IN |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| VSS | U | 14 | IN |
| FCIDBP_L_1_ | U | 15 | BIDIR |
| FCIDB_L_25_ | U | 16 | BIDIR |
| FCIMAUXCMD_L_3_ | U | 17 | OUT |
| FCIMAUXCMD_L_1_ | U | 18 | OUT |
| FCIDB_L_23_ | U | 19 | BIDIR |
| FCIMAUXCMD_L_6_ | U | 20 | OUT |
| VDATA_9_ | V | 2 | BIDIR |
| VSS | V | 3 | IN |
| VAM_1_ | V | 4 | BIDIR |
| VDD | V | 5 | IN |
| VLWORD | V | 6 | BIDIR |
| VGNT_OUT_1_ | V | 7 | OUT |
| VBRQ_IN_2_ | V | 8 | IN |
| VDS1_IN | V | 9 | IN |
| TEST_L | V | 10 | IN |
| VIRQ_6_ | V | 11 | IN |
| CURRENT_CYCLE | V | 12 | IN |
| VIRQ_7_ | V | 13 | IN |
| SCLR_L | V | 14 | IN |
| FCIDB_L_30_ | V | 15 | BIDIR |
| FCIDB_L_27_ | V | 16 | BIDIR |
| FCIDB_L_24_ | V | 17 | BIDIR |
| FCIMAUXCMD_L_2_ | V | 19 | OUT |
| VDATA_8_ | W | 2 | BIDIR |

**Table 14: Chip Pinout by Pin Number**

| name | pin | | direction |
|---|---|---|---|
| VAM_0_ | W | 3 | BIDIR |
| VAM_3_ | W | 4 | BIDIR |
| VAM_5_ | W | 5 | BIDIR |
| VGNT_OUT_0_ | W | 6 | OUT |
| VGNT_OUT_3_ | W | 7 | OUT |
| VDS0_IN | W | 8 | IN |
| VDSX | W | 9 | IN |
| VDSX_D20 | W | 10 | IN |
| VAM_IN_EN | W | 11 | OUT |
| VIRQ_1_ | W | 12 | IN |
| VIRQ_5_ | W | 13 | IN |
| VSS | W | 14 | IN |
| FCIDB_L_31_ | W | 15 | BIDIR |
| FCIDB_L_28_ | W | 16 | BIDIR |
| FCIDB_L_26_ | W | 17 | BIDIR |
| VSS | W | 18 | IN |
| FCIMAUXCMD_L_4_ | W | 19 | OUT |
| VAM_4_ | X | 4 | BIDIR |
| VIACK_L | X | 5 | BIDIR |
| VGNT_OUT_2_ | X | 6 | OUT |
| VSS | X | 7 | IN |
| VIRQ_3_ | X | 13 | IN |
| VSS | X | 14 | IN |
| FCIDB_L_29_ | X | 16 | BIDIR |

# CHAPTER 6 External Buffer Parts

In order to drive the 100 plus pins of a VME bus with the 48/64 ma required for each line, external buffer parts are used for this design. No registered external parts are used in the data path; only buffers and bidirectional buffers. The VME slave handshake (DTACK) logic and bus driver output enable control is implemented externally with three flip-flops and two asynchronous PALs.

.

**Table 15: External Buffer Parts Required**

| Count | Part | Used For |
|---|---|---|
| 4 | 74F623[1] | VDATA[31:0] |
| 4 | 74F623 | VADDR[31:1],LWORD |
| 1 | 74F623 | VAM[5:0], VIACK |
| 1 | 74F125 | VDTACK_OUT,AUX_OUT0/1 |
| 1/2 | 74F244 | VDS0, VDS1, VAS, VWRITE outputs |
| 1/2 | 74F244 | VCLR_OUT, VRESET_OUT, VME_16MHZ_CLK |
| 1 | 74F132 | VDSX, VWRITE_IN, VDS1, VDS0 |
| 5 | 74F14 | various inputs and outputs |
| 1 | 74F04 | buffering for xxxxx_IN output enables |
| 1 | 74F10 | VIACKout logic (all external) |
| 1 | 74F38 | VBERR_OUT, VBSY_OUT, VMYREQ_1, VMYREQ_0 |
| 1 | DELAY | VDXS_DELAY20, VDSX_DELAY25 |
| 1 | 74F74 | AS_OFF detector |
| 1 | 74F112 | two bit grey counter |
| 1 | oscillator | 16Mhz for VME_CLK |
| 1 | oscillator | 50Mhz or 100Mhz for internal clock |
| 1 | PON I.C. | power-on reset logic |
| 2 | PAL | asynchronous logic |
| 28 | Total Discrete Parts | |

1. A 74x623 part is like a 74x245 part with separate enables for each direction instead of a direction and oe control.