How VME is moving to Futurebus+

by Mark Woodbury, Digital Equipment Corporation

January 14, 1992

Introduction

The IEEE Bus Architecture Standards Committee is developing a specification which describes the characteristics necessary to provide VMEbus operability within a Futurebus+ environment. The goal of the defined architecture is to make it very easy from both hardware and software perspectives to port existing applications currently operating in VMEbus systems over to Futurebus+ based systems. The specification does not describe the specific interconnect, but rather defines the minimal set of functions and interface definitions necessary for these goals to be met. It has been designed to allow both low-end and high-end bridges to be built and still conform to the specification by allowing the bridge developer to decide what resources (such as page map registers, dual-port memory size, byte-swap logic, etc) are necessary for the market chosen. It is expect that, while Futurebus+ products will be build to replace the functions currently existing on many VMEbus products, there will still be a need for both bus environments and a bridge between them for many years to come.

Additionally, work is being done within the IEEE to enhance the functionality and performance of the VMEbus and make it compatible with Futurebus+ systems as well. These efforts include VME64 for wider data paths, Source Synchronous transactions (for performance bandwidths above 100Mbyte/sec) and Open Boot PROM for system-independent module configuration and user interfaces.

Features and Limitations

The Bridge spec includes chapters on configuration (CSR and ROM), I/O transactions (Data Port), Interrupt handling (Event Port), Shared Memory (Dual Port) and Byte alignment (Bridge Byte Swapping). These are summarized in Figure 1 and described in some detail in the sections below. It has been important to carefully consider all the relevant functional requirements while limiting the scope of the spec to the simplest necessary subset so that 1) the specification can be usable by a wide application base, and 2) both low-end and high-end products can actually (and economically) be built. The markets considered (and represented by members of the working group) include, but are not limited to, Imaging and Graphics, high performance Real-time, Process control, Storage, Scientific data acquisition, Co-processors and Accelerators, and Network communications for both custom kernel-type and also general purpose multi-tasking Operating Systems.

Some key features and limitations include:

- The Specification is not profile specific
- Data is not in the Cache Coherent domain
- Data is routed via Page mapping control
- Read-Modify-Write transactions are *not* supported across the bridge
- Interrupt transactions supported include:

    Registered - load a register bit to assert an interrupt

    Interlocked - two control flops provide a handshake protocol

    Queued - multiple events may be queued across the bridge

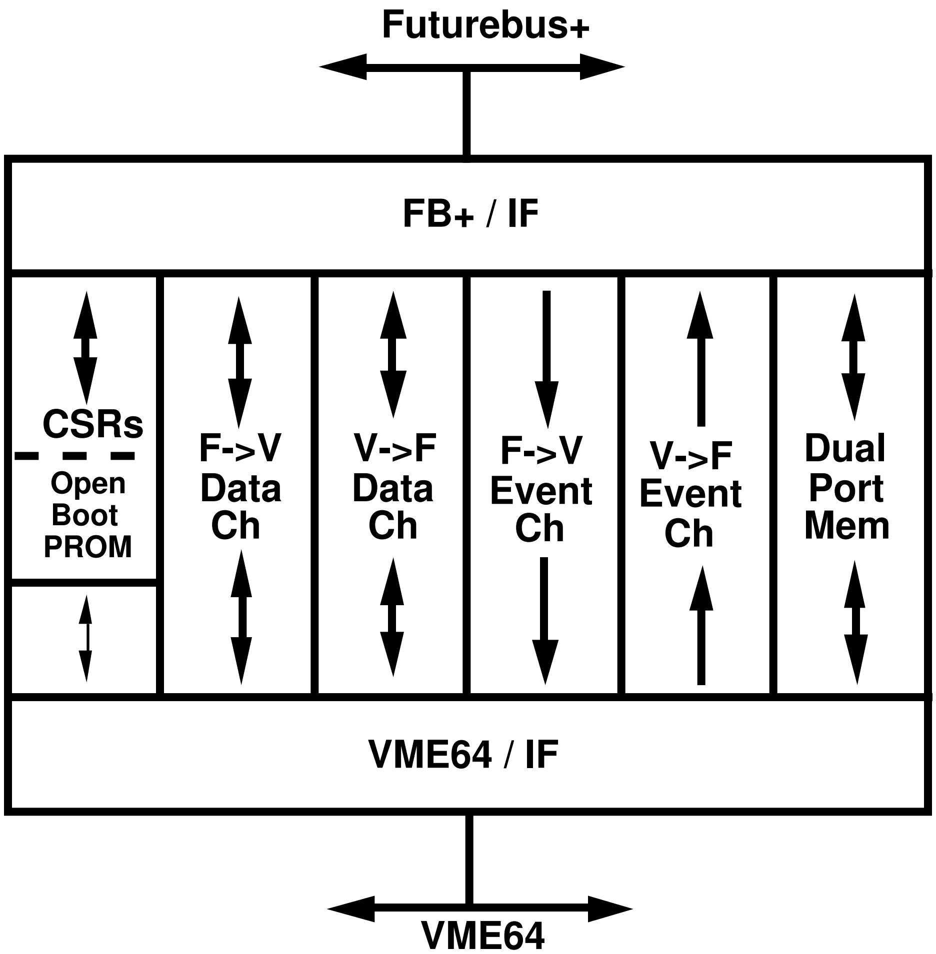

Figure 1. Futurebus+ to VMEbus Architecture

CSR and ROM specifications

The CSR chapter follows the IEEE 1212 specification and highlights Futurebus+ specific and VMEbus specific characteristics. Figure 2 shows all the registers and how the VMEbus is made accessible to the Futurebus+. There are 72 registers specific to the Bridge plus the page map registers for Programmed I/O and DMA transactions (which could be built as a RAM for use with Virtual Memory multi-tasking operating systems).

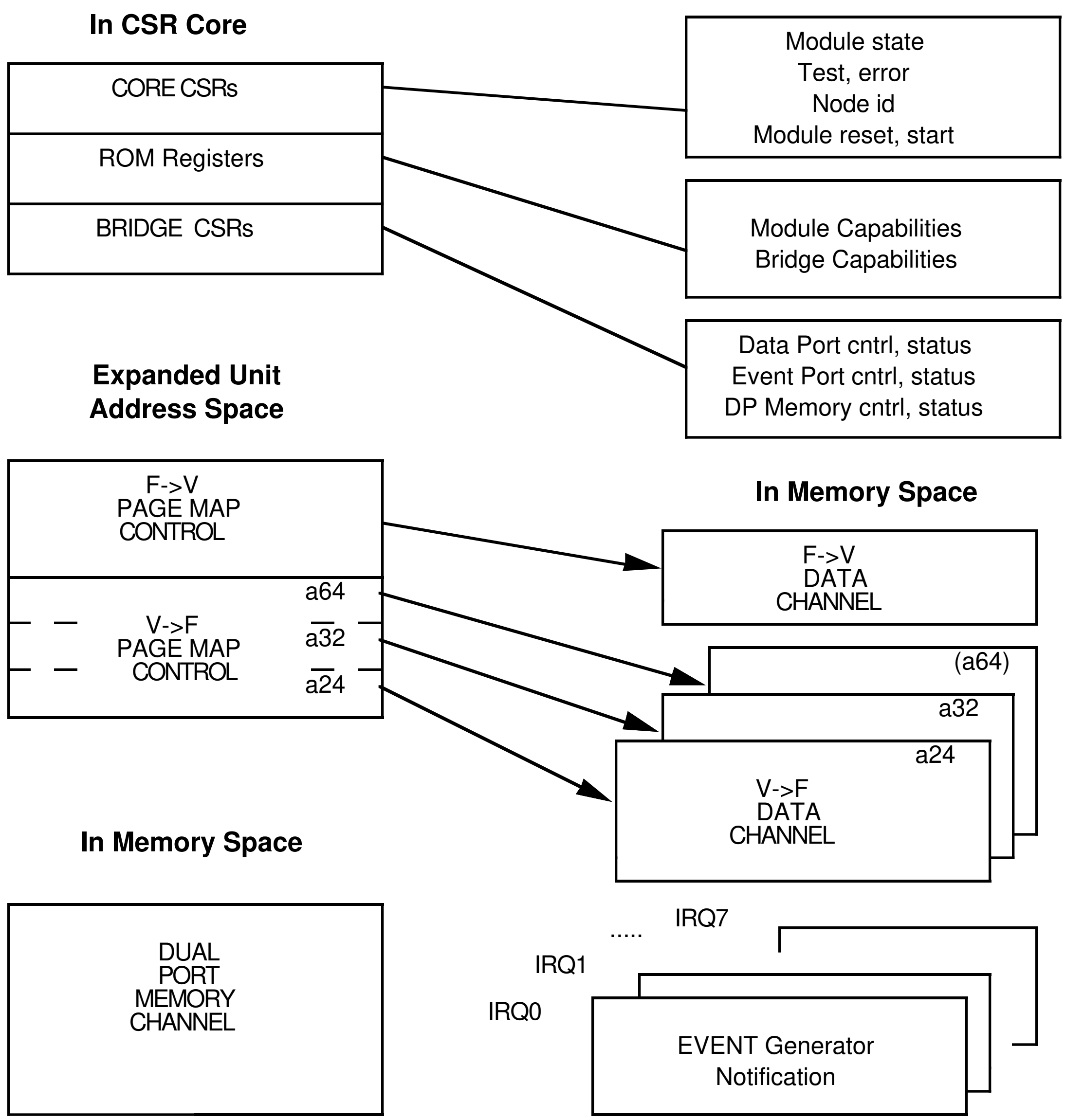

Figure 2: Bridge CSR Map

The ROM specifications describe the Futurebus+ Module characteristics and also describe the capabilities of the specific bridge implementation.

Data Port

The Data port provides a mechanism for transparent access between the two busses and allows physical memory to be sparsely mapped from one bus to the other. Mapping is via pages (which are specified in the ROM and are programmable for size). Page map registers provide control for Futurebus+ originated and VME64 originated transactions. The control is summarized in Table 1.

Futurebus+ to VME64 transactions	**VME64 to Futurebus+ transactions**
Destination Address (high order bits)	Destination Address (high order bits)
VME AM[5:0] code to be asserted	VME AM code to be compared
	Futurebus+ Address type (A64 or A32)
Data Width (max size for each page)	Data Width (max size for each page)
BR[3:0] bus request level	Futurebus+ Priority (3:0) request
Byte Swap mode	Byte Swap Mode
Valid bit (to enable that page)	Valid bit (to enable that page)

Table 1: Data Port Mapping Register Control Summary

Event Port

The event port consists of three classes of interrupts: VME to Futurebus+ event notification, Futurebus+ to VME event notification, and Bridge events. Events can be Register based, Interlocked, or Queued. Notification to the VMEbus of events is via the assertion of an IRQ line. Notification of an event to the Futurebus+ is via a write to an Event register on the Futurebus+ (unique addresses are provided for each interrupt supported). Table 2 provides a summary of the interrupts. User level handling of interrupts is supported by the architecture as each of the 7 IRQ interrupts is allocated a page in memory space (so access is not restricted to priveleged CSR space).

Futurebus+ to VME64 events	**VME64 to Futurebus+ events**
sysreset	sysreset
sysfail	sysfail
acfail	acfail
	event error
	dat error
irq1	irq1
....	....
irq7	irq7

Table 2: Event Types Summary

Dual Port Memory

The Dual Port Memory on the Futurebus+/VMEbus Bridge *may* be used in those applications where either data needs to be shared between modules on the Futurebus+ and VMEbus, or data needs to be quickly written into memory without suffering the inherent connection delays in acquiring the destination bus. The needs are very much application specific, however in general the Dual Port Memory provides a means by which the Bridge can be viewed as memory by devices on both sides concurrently.

Control is provided to: enable VME and Futurebus+ access, specify byte swap mode, and define address window base and bounds (for each bus). The ROM specifies the amount of memory provided on a particular bridge implementation.

Bridge Byte Swapping

Since VMEbus is inherently a Big-endian bus, byte and word operations from a Little-endian processor will, in some cases, yield scrambled data without careful management of that data. The Bridge provides for hardware assisted byte swapping to transparently re-align the data. Even with this hardware assist, care must be taken, since *only* the application has the complete information necessary to properly specify which swapping mode to use. The hardware *does* provide control on a per page basis for performing these swaps without software overhead. The swaping modes supported are listed in Table 3, and refer to the type of data being transferred. Note that the data must, at least, be evenly aligned within it's own size (eg: word swap'd data must be word aligned) for this to work.

A special case (Cycle Specific Mode) is provided to allow the application to specify different data types within the same page. If a byte is written, the bridge will perform a byte swap operation, if a word - word swap, etc. This is very useful for applications which share mixed data structures within the same data field.

Mode	Swap Function	Availability
000	Single Byte	Required
001	Double Byte	Optional
010	Quad Byte	Optional
011	Octet Byte	Required
100	Cycle Specific Byte	Optional

Table 3 : Byte Swap Modes

Conclusions

As Futurebus+ products appear on the market, the initial targets will be towards general purpose, high volume functions such as CPUs, Memory, Disk and Communications. This will leave a gap in systems functionality with the need for custom and niche interfaces. The VMEbus product set currently has these interfaces and, so a Bridge product will fill the functionality gaps. The IEEE Standard insures that the Futurebus+ and VMEbus will maintain this interoperability necessary to meet the full spectrum of customer needs.

About the Author

Mark Woodbury was the architect for the MVME Bus Adapter which bridges the VMEbus onto the TURBOchannel I/O bus for Digital's family of workstations. He is currently in the DMEIC group within Digital Services and is an engineer with their Custom Systems Integration business for Real-Time and Scientific applications utilizing both Ultrix and VMS based platforms. Additionally, he is editor for the IEEE 1014.1 Futurebus+/VME64 Bridge Specification. For further information contact:

Mark Woodbury
Digital Equipment Corporation
Digital Drive, MKO1-1/C15
Merrimack, NH 03054
(603) 884-0850 fax (603) 884-1020
woodbury@solvit.dec.com