From BLT to 2eSST - A Look at the Evolution of VMEbus Protocols

by John Rynearson, Technical Director, VITA
7825 E. Gelding Drive, Suite 104
Scottsdale, AZ 85260
PH: 602.951.8866, FX: 602.951.0720
email: techdir@vita.com
http://www.vita.com/

ABSTRACT: In 1981 when the VMEbus was introduced, the maximum theoretical data rate on the backplane was 40 Mbytes/sec. In 1989 VME64 increased the data rate to 80 Mbytes/sec. In 1996 2eVME provided an increase to 160 Mbytes/sec. In January of 1997, the announcement at the Real Time Computer Conference in Santa Clara, CA provided the opportunity for increase to 320 Mbytes/sec using a new backplane layout refered to as VME320.

This paper takes a look at the evolution of VMEbus protocols from the original BLT to the latest protocol - 2eSST. A discussion of the two newest protocols, 2eVME and 2eSST, is presented along with a look at the enabling backplane technology, VME320.

INTRODUCTION

In 1981 the VMEbus was introduced as a backplane bus architecture for industrial and commercial applications. The data transfer protocols used to define the VMEbus came from the Motorola VERSAbus architecture which owed its heritage to the then recently introduced Motorola 68000 microprocessor.

The VMEbus when introduced defined two basic data transfer operations: single cycle transfers consisting of an address and a data transfer, and a block transfer consisting of an address and a sequence of data transfers. These transfers were asynchronous using a master-slave handshake. The master would put address and data onto the bus and wait for an acknowledgment. The selected slave would either read or write data to or from the bus and then provide a data acknowledge (DTACK*) signal. The speed of data transfers was determined by two things: the physics of the backplane and the speed of the logic used to interface to the backplane bus.

A COOK BOOK APPROACH

Previous to the VMEbus it was not uncommon for backplane busses to require elaborate calculations to determine loading and drive current for interface design. This approach made designs difficult and caused compatibility problems between manufacturers. To make interface design easier and to insure compatibility the developers of the VMEbus architecture defined specific delays based on a 21 slot backplane and mandated the use of certain high current TTL drivers, receivers, and transceivers. The basic idea was to insure interoperability. Interface designers would be assured of an interoperable design if they just followed the rules.

AS*, DSX*, AND DTACK*

The VMEbus data transfer protocol is very straight forward. The master puts addresses onto the bus, delays a minimum of 35 ns, and then asserts address strobe (AS*). For a write operation, the master puts data onto the bus, delays a minimum of 35 nsecs, and then asserts one or both of its data strobes (DS0* and/or DS1*). All slave cards on the bus monitor the addresses. Each slave is set up to decode a unique address. The assertion of AS* tells the slave that the address is valid. In a write cycle the selected slave must then read data off the bus. The assertion of data strobe tells the slave that data is valid on the bus and can be strobed into memory. The slave then asserts data acknowledge (DTACK*) to signal that the data has been captured.

A timing diagram for such a data transfer is shown in Figure 1.

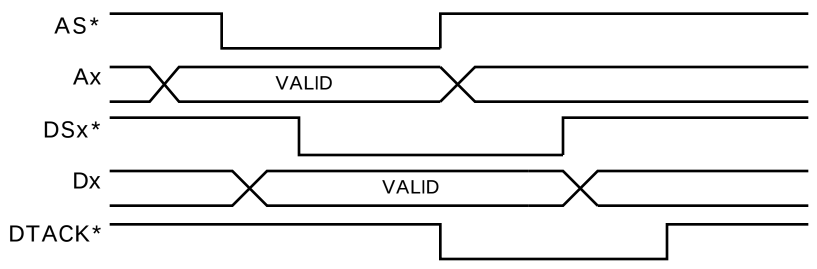

Figure 1 - Write Data Transfer

TIMING

The timing delays in the VMEbus standard are based on the physics of the 21 slot backplane. It takes a certain amount of current to drive a 5 volt TTL signal to ground. The amount of current is based on the load that the driver sees. On a fully loaded 21 slot backplane a driver will see an impedance of approximately 20 ohms. The amount of current required to drive a bus signal to ground is much greater than a TTL gate can provide. Therefore, a transition from low to high or high to low is not achieved on the first past of the wavefront down the bus. The wavefront must reflect off of the termination resistors and then progress back down the bus. Because of this effect the VMEbus operates as a reflective wave backplane. That is, the transition from one logic level to another is not guaranteed on the first pass of the wavefront, but requires a reflection to make the transition complete. The delays in the VMEbus standard make sure that signals have reached steady state before being strobed.

THEORETICAL SPEED

"How fast can I transfer data?" Real time applications, unlike desktop ones, must meet specific time requirements. When an event happens, a response must take place within a specific period of time. Otherwise the requirements of the application are not met. When the VMEbus was first introduced, backplane speed was a key factor in overall system performance. Backplane performance is determined by two things: the width of the data path and the maximum bit transfer rate. The VMEbus when released specified data transfers of 8, 16, and 32 bit widths. The amount of time for a single transfer was the same whether 8 or 32 bits was transferred, but the rate of transfer is 4x for 32 versus 8 bits. The transfer rate is determined by the speed of bus drivers and receivers and the built-in delays that are part of the VMEbus standard. The standard notes that AS* must be driven high for a minimum of 40 nsec before it can be driven low again. Additionally, AS* must be driven low for a minimum of 40 nsec before it can be driven high. Therefore the fastest AS* cycle would total 80 nsec if driver and receiver delays could be ignored. Since driver and receiver delays must also be taken into account, an additional 10 nsec for driver delay and 10 nsec for receiver delay are added to get a minimum cycle time of 100 nsec per data transfer. While current day drivers and receivers have sub 10 nsec delays, rounding up to 10 nsec give a nice figure of 100 nsec per transfer or a 10 Mhz transfer rate. This transfer rate for 32 bit data gives a maximum transfer rate of 40 Mbytes/sec.

PRACTICAL TRANSFER RATES

While the theoretical data transfer rate for 32 bit transfers is 40 Mbytes/sec, what rates are achievable in practice? Overall system performance is a function of hardware design and software implementation. A 40 Mbyte/second data rate implies an AS* cycle time of 100 nsec per 32 bit transfer. Even for today's high performance processors 100 nanoseconds is not a lot of time to transfer data while multiplexing other tasks. Therefore, practical data rates for 32 bit transfers of 30-35 Mbytes/sec are only possible by paying careful attention to hardware design, software overhead, and system loading.

THE NEED FOR SPEED

In 1989 the technical committee of VITA met in Paris to review the technical needs of the VMEbus specification for the coming decade. At that meeting, John Peters, Performance Technology, revealed a method that they had used to multiplex 32 bits of data onto address lines to achieve 64 bit transfers during a VMEbus block transfer. The concept was christened VME64 and the new block transfer protocol was called multiplexed block transfer or MBLT. In addition it was suggested that A64 addresses could be generated by multiplexing addresses onto the data lines. The addition of MBLT to the repertoire of VMEbus protocols was easy due to the use of address modifier codes. Address modifier codes occur on every VMEbus cycle and qualify each cycle by specifying its type.

The addition of 64 bit transfers provided a dramatic 2x performance enhancement and thwarted the need for a new bus architecture.

MORE SPEED

In order to complete the development of the VME64 specification certain extensions were moved to the VME64x specification. One of those extensions was a new protocol that was originally named 2eBLT and was latter changed to 2eVME. The "2e" refers to two edge transfers. Traditionally, VME protocols have used four edge handshakes for a single data transfer as shown in Figure 2. The assertion by the master of the strobe at edge 1 lets the slave know that data is valid. The slave acknowledges with edge 2. The assertion of edge 2 lets the master know that the slave has responded so that it rescinds its strobe at edge 3. The slave then finishes the cycle by rescinding its signal at edge 4. Thus a single data transfer requires four edges.

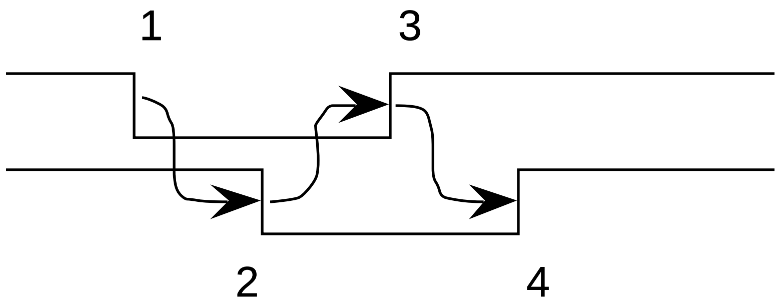

Figure 2 - Four Edge Handshake - Single data Transfer

However, it was realized that another 2x in performance enhancement could be gained by transferring data on both the rising edge and the falling edge of the transfer signal. See Figure 3. The master signals data valid on the falling edge of the strobe at 1 and the slave acknowledges data received by asserting its strobe at edge 2. A single transfer is accomplished with two edges. The next transfer is signaled with the rising edge at 1 and the slave acknowledges with edge 2. The effective data rate is 20 megatransactions per second. The new protocol was named 2eVME for two edge VME and is specified in the VITA 1.1-1997, VME64x draft specification.

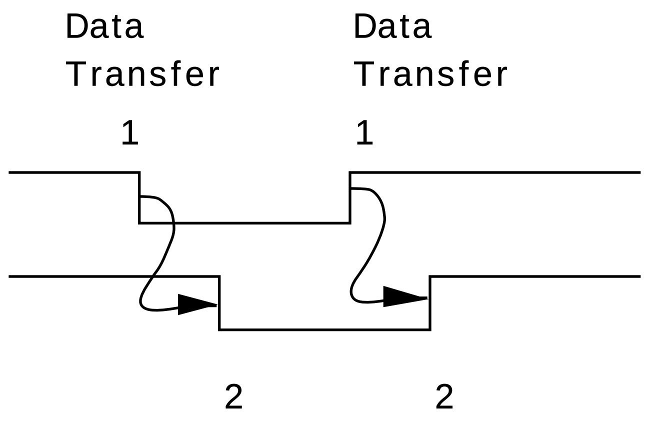

Figure 3 - Two edge transfer

<u>2eVME</u>

The 2eVME protocol adds four important features to the VME64 and VME64x architectures. These features are:

1. Doubling of the peak block transfer rate from 80 Mbytes/sec to 160 Mbytes/sec.
2. Master and slave terminated transfers.
3. Extended address modifier codes.

Part of the 2eVME protocol is an extended addressing phase to allow for additional address modifier codes. Original VME allowed for 64 address modifier codes. Since there are only a few unassigned address modifier codes left, an extended address modifier (XAM) coding scheme is used. AM Code 0x20 is assigned for 6U 2eVME transfers and AM Code 0x21 is assigned for 3U 2eVME transfers. The eight LSB (least significant bits) of the address field A[7:0] are used to carry the extended address modifier code information during the first address phase. With 8 bits, 256 additional address modifier codes are available for each of the 6U and 3U 2eVME transaction sets.

Two extended address modifier codes each are allocated for 6U modules (A32 and A64) and for 3U modules (A32 and A40).   The usage of supervisory/user mode and program/data mode are no longer used since A64 provides a sufficiently large address space.

2eVME transfers may be terminated by either the slave or the master. During the second phase of the address cycle, the maximum number of data transfers is transmitted to the slave. This count can range from 0 to 256 transfers. Slaves can terminate the transfer at any time by asserting RETRY* and then asserting BERR* in the case of 6U modules and RESP* and then BERR* in the case of 3U modules. This termination method tells the master that the slave is done with the transfer.

Slaves can also suspend a data transfer for resumption at a later time by asserting RETRY* and then toggling DTACK* in the case of 6U modules or asserting RESP* and then toggling DTACK* in the case of 3U modules. The suspend termination tells the master that the slave is stopping this transaction, but is expecting a resumption of that transfer at a later time. Suspended transactions always occur on an even beat count. The 2eVME protocol is part of the new VITA 1.1-1997, VME64x draft standard, which has been submitted to ANSI as a proposed American National Standard.

A NEW BACKPLANE

At the January 1997 Real Time Computer show in Santa Clara, CA. an astonishing announcement was made. Drew Berding, Arizona Digital, displayed a 21 slot VMEbus backplane that could transfer data at a 320 Mbyte/second data rate. His demonstration was even more dramatic because one of the modules was on a standard extender board. Scope traces showed the signals to be very clean even though signals were running from slot 1 to slot 21 and then through an extender board in a fully loaded system.

Much speculation ensued as to how the backplane was constructed. While many ideas were put forth, the method employed by Drew was both elegant and straight forward. A standard backplane trace goes from slot 1 to slot 2 to slot 3 and so on through slot 21. In the VME320 backplane each slot trace is wired directly to slot 11 in a basic star configuration. So that a signal from slot 1 to slot 2 goes from slot 1 to slot 11 and then back to slot 2. Wiring the backplane in this manner changes the backplane signal characteristics. Instead of looking like a transmission line, the VME320 backplane looks like a lumped capacitance. As a result signal lines are monotonic, noise is reduced, and data transfer rates can be effectively increased.

2eSST

To take advantage of VME320 backplane technology, the VITA Stamdards Organization (VSO) set up a task group early in 1997 to begin work on a new protocol. The task group decided to specify a synchronous protocol that would provide data transfer rates of 320 Mbytes/second.

The 2eSST protocol, is based on the 2eVME protocol. The main exception to this is that during its data phases, 2eSST is a source synchronous protocol. No acknowledgment is expected from the receiver of the data. Hence, the theoretical performance of 2eSST is limited only by the skew between receiver and transmitter of data. The result is a protocol that as currently defined doubles the theoretical bandwidth of VME to 320Mbytes/sec. The protocol can be broken into three main phases: address broadcast, data phase and termination.

The address broadcast phase for 2eSST is identical to the address broadcast phase for 2eVME. However, the data phase is synchronous rather than asynchronous.

Traditional VME utilizes a handshake protocol whereby data strobes (DS1* and DS0*) are acknowledged by DTACK* which then allows the data strobes to be removed which in turn allows the DTACK* to be removed. Once DTACK* is deasserted, a new cycle can begin.

Traditional VME protocol requires four delays through the drivers, backplane and receivers plus the settling time of the backplane. 2eVME protocol improves upon this by using both edges of DS1*, DS0*, and DTACK* to qualify data. Throughput is doubled, but performance is still limited by the requirement for acknowledgment from the receiver of data.

In contrast, after the address phase, the 2eSST protocol sends the data and strobe and does not wait for any acknowledgments. Therefore, data can be sent at much higher bandwidth. Both edges of the strobe are used: falling edge for odd data beats and rising edge for even data beats. For write transfers where the master is the transmitter of data, DS1* is used to qualify the data. For read transfer where the slave is the transmitter of data, DTACK* is used to qualify the data.

A consequence of the source synchronous nature of 2eSST is that the receiver of data no longer has the ability to throttle the data transfer. In traditional VME block transfer reads, the master (the receiver of data) could throttle by controlling its assertion/deassertion of DS0*/DS1*. Writes could be throttled by the slave (the receiver of data) by controlling DTACK*. In 2eSST, only the transmitter of data can throttle data. This implies that the 2eSST receiver (master during reads, slave during writes) must be able to accept the data at the rate it is sent. This is sometimes referred to as the "fire-hose" approach to sending data.

While the receiver of data during a 2eSST transfer has no ability to throttle data transfers during the data phase, it does have the ability to stop the transaction should its on-board resources be unable to sink any further data. Both master and slave have the ability to stop the transfer at any even beat during reads or writes. When the master terminates the transfer, its is referred to as a "Master Termination". When the slave terminates the transfer it is referred to as a "Slave Suspend". The slave suspend operates as a retry mechanism, in that the master may initiate another transaction following the suspension to continue with the data transfers at the next data beat.

Termination only on even data beats helps to achieve two of the objectives of 2eSST: it minimizes design complexity while maximizing throughput. By not allowing termination at odd beats, 2eSST state machines need not design for the special case where the transfer stops with the strobes in the asserted position necessitating a "clean-up" dummy cycle to return the strobes to the negated state. In addition, the number of test conditions a design must be verified under is cut in half. By removing the potential for a dummy cycle, and by removing the requirement to monitor for terminations on every cycle, the state machines will also be able to run faster. As with other protocols, a slave error termination is also included with 2eSST. This permits the slave to relay some error condition back to the master.

VITA 1.5-199x, 2eSST is the draft standard for the 2eSST protocol. The task group expects to submit this standard to ANSI during 1998.

SUMMARY

Over time the VMEbus has evolved to meet new application requirements. The key to this evolution has been the address modifier codes and the decision of the developers of the VMEbus specification to maintain interoperability with each new addition. With the advent of VME320 backplane technology in 1997, VMEbus backplane performance has been enhanced 8x since VME's introduction in 1981.

REFERENCES

(All standards are available from the VITA office. Some draft standards are available in PDF and may be downloaded from the VITA web site, http://www.vita.com/)

1. ANSI/VITA 1-1994, VME64 Specification
2. VITA 1.1-1997, VME64x
3. VITA 1.5-199x, 2eSST
4. http://www2.bustronic.com/bustronic/VME320/vme320pres.html