SLAC - PUB - 4414
September 1987
(I)

HIGH PERFORMANCE DATA BUSES—PROGRESS AND EVOLUTION*

David B. Gustavson
Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94305

Abstract

In 1987 four new 32-bit computer backplane buses joined Fastbus (ANSI/IEEE Std. 960-1986) as IEEE standards. They are Futurebus (IEEE Std. 896.1), VME (IEEE Std. 1014), NuBus (IEEE Std. 1196), and Multibus-II (IEEE Std. 1296).

This paper compares and contrasts these buses, discusses their strengths and weaknesses, and considers possible directions for future improvement.

Some of these buses have reached fundamental limits. How can they be improved? Can they be extended in ways which are backward compatible with present buses, or will they be abandoned in favor of new designs? What direction will new bus designs take?

Introduction

Industry standard general-purpose computer buses have been around for over a decade. Computer buses, of course, are much older than that, but I am referring to the kind of bus which forms a standard interconnection point for which multiple manufacturers build compatible and interchangeable interfaces.

Some of these buses are manufacturer proprietary, like DEC's Unibus, and others began as proprietary buses but evolved into official ANSI/IEEE/IEC standards, like S-100 (ANSI/IEEE Std. 696-1983) and Multibus (ANSI/IEEE Std. 796-1983). 696 and 796 began as 8-bit microprocessor buses, started by a single vendor, but they filled a great need and industry adopted them in large numbers of applications. During the standardization process they evolved into versatile 16-bit buses. In these cases, significant technical improvement was incorporated during standardization.

Standardization tends to be a slow process because of the time it takes to reach a consensus from initially widely disparate viewpoints; the problem is aggravated by rapid turnover of the committee membership and by a general lack of resources compared to the magnitude of the job.

Recognizing the time required for this process, and the desirability of having a small number of standards in any particular category, the IEEE Computer Society's Microprocessor Standards Committee made an early start at defining a 32-bit bus, to be called "Futurebus". It was clear that a standard 32-bit bus would be needed eventually, and the hope was to design one which did things right, one standard which would be adopted by nearly everyone.

It didn't turn out that way, for various reasons. The process of defining a new standard from nothing, in a committee, is much slower than the process of fixing up a few problems and adding a few features to an already existing industry standard. The requirement of backward compatibility is a strong stabilizing influence, and greatly speeds convergence to consensus. Furthermore, industrial design teams composed of a small number of individuals in close communication can work much faster than a standards committee composed of a varying number of individuals communicating occasionally. Given the nature of the process, I think it is amazing that Futurebus ever completed at all, and the story of its evolution would make an interesting thesis in sociology.

Impatience with this process, along with historical accident and incompatible goals and visions, caused several 32-bit buses to appear at about the same time. The first to finish was Fastbus, developed in a more coherent atmosphere under the Nuclear and Plasma Physics Society by an organization with experience from the earlier development of CAMAC (ANSI/IEEE 583). Fastbus's goals were larger than those of the other buses, due to the data-acquisition/highly-parallel-processing environment for which it was developed. Fastbus is also very processor independent, ignoring the peculiar needs of any particular microprocessor chip and concerning itself only with the general architectural problems of managing large systems at low cost. Fastbus goes so far as to ignore byte addressing altogether, for example. Modules can transfer bytes if they like, but it is discouraged, and the significance of bytes within words is left to the particular application to define. Though Fastbus is also suitable for small systems, they imposed no constraints on the bus protocol design process.

The later buses were largely based on the assumption that microprocessors would be their main users, and that a single backplane would be the normal high-end limit to the system size.

Everybody has 32 bits

32 bits has become accepted as the magic number needed for today's microprocessors, so most new buses at least provide for a 32-bit expansion capability. Even the old 696 and 796 buses are in danger of getting 32-bit extensions. The buses I will discuss in this paper, however, were mostly designed as 32-bit buses from the start. I will present them in increasing order of my own preference. I will omit the VSB (IEEE 1096, IEC 822) because it is intended as a subsystem bus and has limitations (e.g. the maximum number of masters) which make it unsuitable for general backplane use. I also omit the "IBM" PC bus (IEEE P996) which is to incorporate a 32-bit version, which as of this writing looks more likely to be the NuBus than the IBM PS/2 Micro-Channel! The PC/AT bus may be used as an 8–16-bit I/O bus in this 32-bit implementation. Alternatively, this combination might be renumbered as 1196.2 to emphasize its relationship to NuBus.

VME (IEEE 1014)

VME began as the Motorola 68000 microprocessor bus, expanded slightly for use on a backplane. It was expanded to 32 bits during standardization by using a second connector for 8 additional address and 16 additional data lines. Its history is evident upon inspection; the arbitration system relies on daisy chains and cannot provide true fairness (equal access opportunity) for more than four processors, and the interrupt system is basically the old style used in single-processor systems, with minor generalizations.

The signalling technology is TTL, and the bus is asynchronous; this is a touchy point, as asynchronous systems need clean signal edges, and TTL buses cannot deliver them reliably. There are far too few ground pins in the connector, incompatible with any model of driving transmission lines in the backplane, and causing "ground shift" which acts like noise on the timing lines. These problems must be solved by suitable delays or low-pass filters, whether explicit or implicit, which provide the time needed for the signals to clean up.

VME is non-multiplexed, using two 96-pin connectors to provide enough lines for separate address and data. There is no Geographic Addressing mechanism, and therefore no CSR (Control and Status Register) Space, needed for automatic self-configuration. VME supports 20 module positions. The bus is Big-Endian[1] (increasing byte addresses from left to right or high-order to low-order within a word), partially justified (8- and 16-bit transfers use the low-order 16 data lines), with a throughput of about 19 Mbytes/sec for single transfers and

* Work supported by the Department of Energy, contract DE-AC03-76SF00515.

21 Mbytes/sec during blocks[2].

VME is very popular, particularly the 16-bit subset, and works well if the system limits are not pushed too hard. It would not be a good choice for a crate full of 32-bit processors, but it is convenient and economical for single processors and for I/O cards. Because of its popularity and availability, most other buses offer or plan to offer some interface mechanism which allows the use of VME cards or crates for initial I/O needs, which helps them avoid the chicken/egg problem during initial market development.

MultiBus-II (IEEE 1296)

Multibus-II is a 32-bit multiplexed, 10 MHz synchronous bus, which uses Fastbus-style arbitration and has a form of Geographical Addressing based on T-pins which can be used for automatic initialization. It has a complex structure, incorporating memory, I/O, Initialization (CSR) and Message Passing spaces and protocols. It uses TTL signalling.

Synchronous buses pay a synchronization penalty averaging half a clock period, assuming the on-board processor runs on its own clock (very probable, now that processor clocks run far faster than 10 MHz). That is, when the processor decides it wants access to the bus, on average it will have to wait half a bus clock cycle to become synchronized to the bus, a delay which does not occur with asynchronous buses. On the other hand, synchronous buses are supposed to be simpler to design for and simpler to debug, though the complexity of this one largely cancels (or perhaps completely overwhelms) that benefit.

The great benefit of synchronous buses is that the clock provides the time needed for poor signals to settle before they are looked at; this is extremely important for TTL-based signalling, which is doomed to have very poor transmission line behavior.

The bus is Little-Endian, partially justified (8- and 16-bit transfers use the low-order 16 bits), with speeds of about 20 Mbytes/sec for either single or block transfers, and a limiting speed of 40 Mbytes/sec[2]. It is a single-segment design, and uses IEEE 1101 mechanics, a common mechanical standard (Eurocard-related) shared by 896.1, 1196, and 1296. Multibus-II supports 20 module positions.

Multibus-II is backed by enormous corporate resources, and seems certain to achieve some level of success as a result. The VLSI support chips which such resources make possible are certainly essential for any economical implementation.

NuBus (IEEE 1196)

NuBus is a 32-bit 10 MHz multiplexed synchronous bus, with Fastbus-style arbitration which is strictly fair, i.e. all modules get an equal chance at the bus. It has Geographical Addresses encoded in the backplane at each connector position, allows only 16 boards on a backplane, has CSRs as a defined subset of the single memory-style address space, and uses TTL signalling.

NuBus has a long evolutionary history dating back to MIT and passing through Western Digital and Texas Instruments. It is remarkable (especially for a standard evolved through a committee) in its sparcity of mechanism. Virtually every feature which has survived serves more than one purpose. Some of the benefits are not obvious—for example, the block transfer mechanism seems limited at first because it restricts the lengths and starting addresses of transfers. However, that restriction eliminates a whole class of problems (what happens when a block transfer tries to cross module boundaries?) without sacrificing significant performance. Its simplicity results in simple and inexpensive interfaces, practical even without VLSI, and gives the user a chance of understanding what is going on.

NuBus was adopted by Apple for the Macintosh-II open architecture machine. A PC-style board format was defined for this purpose in addition to the original IEEE 1101 mechanics. There are a few other manufacturers using it as well, and soon there will be many as momentum toward the Mac-II increases. As mentioned above, it is the probable choice for the IEEE P996 "PC" 32-bit bus!

It is Little-Endian, non-justified, with transfer rates of 20 Mbytes/sec for single or block transfers, and a limiting rate of 37.5 Mbytes/sec[2]. It has a single-segment design.

Futurebus (IEEE 896.1)

Futurebus is a 32-bit multiplexed asynchronous bus, using Fastbus-style arbitration with a distributed control mechanism. There is no central logic such as the clock driver used by Multibus-II or NuBus, or the arbitration timing used by Fastbus. Some complexity was added in the process of avoiding central logic elements and technology dependencies, and some performance penalty was also accepted. Nevertheless, Futurebus is an elegant high-performance bus. It has Geographical Address pins encoded in the backplane at each socket, supports 21 modules per backplane, and has CSRs allocated at defined locations in memory-style address space.

Futurebus is non-Endian (it provides byte addressing without specifying the order of byte addresses within a word), non-justified, with a throughput of about 25 Mbytes/sec for single, 44 for block, and an ultimate limit of about 95 Mbytes/sec[2]. It was designed as a single-segment system, but efforts are underway to allow multiple segments to communicate. It uses IEEE 1101 mechanics.

Futurebus can claim two major breakthroughs. First, its signalling technology is based on devices which did not exist until Futurebus specified them, namely BTL (Backplane Transceiver Logic) chips which provide small signal swings of about one volt, with controlled rise and fall times, and (most importantly) very low capacitance. BTL is far superior to TTL for signalling on buses. The small signal swing makes proper termination feasible at reasonable power levels, and the bus impedance is improved by the low BTL capacitance. Though 10K ECL (as used by Fastbus) is about as good, BTL avoids the need for ECL's negative supply voltages by signalling between +1 and +2 volts.

The second breakthrough is Futurebus' support of distributed shared-memory cache systems. Cache memories are becoming increasingly important as processor speeds increase, and can greatly reduce bus bandwidth requirements for a given level of performance. For example, with a proper design a processor can be testing a system semaphore in a loop without disturbing the other users of the bus, if a copy of the semaphore is in the processor's own cache memory. However, for this to be valid it is necessary that any other processor's write to that semaphore be detected by the cache, so that the new value will be seen. Futurebus supports several schemes for doing this. One cost of doing this is that all address cycles are broadcast, and all modules participate in the handshake. This slows the system a little, but the benefit seems worth that price. This would be too costly on long buses, such as Fastbus's cable segments, but on a backplane it is acceptable.

Futurebus has a very general broadcast protocol which is self-timing, as every module participates in the handshake for each cycle. The arbitration control mechanism also uses a generalized handshake of this sort which results in technology independence at some cost in complexity[3]. Futurebus generalized from the Fastbus protocols, optimizing for the single backplane environment, and escalating the principle of technology independence.

Fastbus (IEEE 960, IEC 935)

Fastbus is a 32-bit multiplexed asynchronous bus, using fast parallel arbitration (centrally timed) including optional fairness and priority and priority-deferring strategies. The protocol also includes synchronous (pipelined) block transfers, which can reach the ultimate speed limits of the bus medium by eliminating the delay of waiting for the handshake to arrive at the sender.

Fastbus supports 26 modules per backplane, with Geographical Addresses encoded at each backplane position. CSRs are in a separate address space, designed so that it is easy to access any CSR in a particular module given any address to which that module responds. A fundamentally important feature of Fastbus is its support of multiple segments, including long flexible cable segments. The protocol was designed with segment interconnection in mind, and with the constraint that the same protocol should be used on cables and on backplanes. Thus it is possible to have "modules" such as large computers (which would not fit in a normal module in a crate) attached to a cable segment.

This flexibility has major architectural implications. Fastbus allows arbitrarily complex interconnection topologies, potentially even allowing alternate routes between two points in a connected system. Tree, star, ring or jumbled connections are allowed. A versatile broadcast addressing mechanism allows broadcasts along paths or to subtrees. The CSR mechanism and address allocation mechanism are designed for ultimate expandability by using a special "Secondary Address" data cycle which effectively removes all limits which the 32-bit address field might have imposed.

Fastbus specifies a complete mechanical system, with air or liquid cooling. It provides excellent power and ground distribution, and a convenient method for attaching I/O cables to the back of an auxiliary area on the backplane so that cabling need not be disturbed when exchanging modules.

Fastbus is non-Endian, supporting only word addressing and word transfers, and thus is non-justified as well. Byte addressing which may be desired by particular processors or applications can be handled in their interface to Fastbus. Throughput is about 25 Mbytes/sec for single word transfers, 55 for typical blocks[2]; ultimate limits are in excess of 200 Mbytes/sec (an existing implementation routinely operates at 160[4]).

What are the Limits, What's Next?

The ultimate speed limits are set by bus width and by the bandwidth of the signalling system, which is limited by dispersion, reflections and noise. In addition, the handshake (or clock, if synchronous) causes limits due to propagation delay (time for the data to go one way and the acknowledging handshake to come back); protocols may be inefficient, wasting bus time; and bus skew can be important. Skew is the difference in transmission time on one of the parallel lines compared to another. Part of the skew is caused by actual differences in the lines due to tolerances or imperfections, and part is due to variations in the transceiver threshholds, drive capability, and delay. The Fastbus throughput limit for pipelined transfers is almost entirely due to skew, for example.

How can we beat these limits? First, we always gain with short, fat buses. The next generation may use 128 bits, matching common microprocessor chips' cache line sizes. For dedicated applications short buses may be acceptable. For high performance systems they may be essential.

Skew can be reduced by careful design, tight specs on bus transceivers, or trickery. For example, most transceivers are available in octal packages. If the bus is divided into 8-bit pieces each with their own timing lines and using only transceivers from a single chip (inherently well-matched), skew may be effectively reduced. Another approach would be to use self-clocking data, which is what modern multitrack tape drives do. Each bit's data stream would be received separately, and at the end of the block the pieces would be put together coherently in the controller. This implies VLSI circuitry which is much faster than normal bus signalling rates in order to be practical.

Protocol delays can be reduced in two general ways. Split transaction buses effectively send write-only packets, so there is no waiting on the bus for the response. For example, a packet is sent to a memory requesting certain data and telling the memory to whom to send it. Then the bus is used by others, until the memory has retrieved the data and uses the bus to send it to the requester. The packets can be implemented on present bus structures, with handshake and status as on a normal write, but for ultimate performance they should become more like local-network packets with no handshake delays; i.e., the handshake is replaced by a higher-level acknowledge protocol.

The other way to reduce protocol delays is pipelining. If the sender pumps the data out without waiting for acknowledging handshakes, the propagation delay does not limit throughput. There are several partial implementations which might be useful. For example, the parity bits normally show up after some logic delay, and present protocols wait to signal data presence until parity is stable. In principle, however, parity could be pipelined so that it always arrives later, perhaps with its own strobe (unless a fixed delay can be depended on).

In addition to the 2-line (DS/DK) pipelining used by Fastbus, an N-line polyphase scheme has been developed by Keith Britton. The theoretical speeds are probably identical, as both are limited by data bandwidth and skew, but control logic may be simpler for one or the other. Interesting problems are related to the changeover from addressing (establishment of connection) to data flow and from one bus owner to another.

Another idea which shows promise is the active backplane. The bus could be standardized at the module connector, and internally contain all the transceivers and transmission lines. One might purchase a more or less expensive bus logic assembly depending on the system requirements, without having to know in detail what is inside. This approach could result in carefully tuned buses with very low skew and good signalling properties; real buses of the present sorts suffer greatly from the compromises needed to accommodate variable loading as modules are inserted and removed in various places. An active backplane could more easily accommodate power-on insertion and removal of modules as well. Possibly the bus logic could take on some of the burden of the board designer, incorporating some functions common to most boards—some decoding, standard control registers and mechanisms, a message transfer system, packet buffers, error correction, maybe even a cache interface.

So what's the next step? There is now a study group sponsored by the IEEE Computer Society's Microprocessor Standards Committee, called the Superbus Study Group, convened by Paul Sweazey of National Semiconductor, tel. (408) 721-5860. This group will consider goals and the feasibility of a new higher-performance standard (about 1 Gigabyte/sec) to be completed in the 1990s. Designers should not wait for the Superbus, as it will require technology not soon available (and the committee may never agree on its specs!), but should move to Fastbus, Futurebus, or NuBus now. They are the practical standards for the next decade.

References

1. D. Cohen, "On Holy Wars and a Plea for Peace," *IEEE Computer*, Vol. 14, No. 10, October 1981, pp. 48-54.
2. Paul L. Borrill, "MicroStandards Special Feature: A Comparison of 32-Bit Buses," *IEEE Micro*, December 1985, pp. 71–79. I chose the times corresponding to 50 ns access time.
3. D. M. Taub, "Arbitration and Control Acquisition in the Proposed IEEE 896 Futurebus," *IEEE Micro*, August 1984, pp. 28–41.

   D. M. Taub, "Improved Control Acquisition Scheme for the IEEE 896 Futurebus," *IEEE Micro*, June 1987, pp. 52–62.
4. E. Siskind, "Experience with a Fastbus Based Data Acquisition System for Imaging Coronary Arteries," *IEEE Transactions on Nuclear Science*, Vol. NS-31, No. 1, February 1984, pp. 230–235.