SLAC – PUB – 3844
December 1985
(I)

# AN INTRODUCTION TO THE FASTBUS*

DAVID B. GUSTAVSON

*Stanford Linear Accelerator Center*
*Stanford University, Stanford, California, 94305*

## ABSTRACT

The Fastbus is a modular data bus system for data acquisition and data processing. It is a multiprocessor system with multiple bus segments which operate independently but link together for passing data. It operates asynchronously to accommodate very high and very low speed devices over long or short paths, using handshake protocols for reliability. It can also operate synchronously with pipelined handshakes for transfer of data blocks at maximum speed. The goals, history and motivation for the Fastbus are summarized briefly. The structure of the Fastbus system is described in general and some details of its operation are introduced.

## HISTORY AND GOALS OF THE FASTBUS DESIGN

In 1975 several physicists active in high energy elementary particle physics asked the US NIM (Nuclear Instrumentation Modules) committee to consider development of a standard data acquisition system which would meet the needs of future physics experiments and which would also be of general utility. The NIM committee established a study group to examine the needs and determine objectives to be met by such a system. In 1977, a committee was formed to begin the design of a system which would meet these goals. The design and prototyping efforts were supported by the U. S. Department of Energy.

The design committee essentially completed its work in 1983, publishing the Fastbus specification as Department of Energy report DOE/ER-0189. Fastbus became ANSI/IEEE Standard 960 in 1984. As of this writing, Fastbus has reached routine commercial production; catalogs of off-the-shelf Fastbus equipment are available from several vendors. Committee work continues, now mainly focussed on standards for software supporting Fastbus.

The goals of the Fastbus design were:

> Highest possible speed, in order to handle the high data rates encountered in complex particle detectors, and to reduce the temptation to design custom systems for each application.

Lowest possible cost, because huge volumes of electronics are required in modern experimental physics and any unnecessary cost is multiplied by a large factor.

Modular construction, so that useful standard building blocks can be developed which can then be connected in various ways to meet the needs of various projects.

General utility, so that commercial use outside the physics community will raise vendor volume and lower user costs. No special physics features which interfere with general-purpose use in other markets.

Complete system design, including: power distribution and heat removal adequate for large high-speed systems; diagnostic facilities to speed discovery, location and correction of faults; operating and maintenance aids such as rear access for cabling, machine-readable module identifiers, standardized control and status registers.

Multiple-processor organization, so that many low-cost computing elements can be brought to bear on parts of the data acquisition and analysis problem.

Segmented organization, so that the multiple processors will not be overly limited by the bus bandwidth. Segments of the bus should operate independently, joining together automatically as needed for passing data.

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

Invited paper presented at the BUSCON, San Jose, California, January 14-16, 1986.

Asynchronous operation, so that slow and fast modules can be mixed in the system, and so that the system speed can increase easily as technology advances.

Synchronous operation, so that data transfer can occur at the full bandwidth capability of the transfer medium when ultimate performance is needed.

Uniform protocols, so that no protocol translation is needed as data flows from segment to segment. The Fastbus uses cable segments to join segment interconnect modules residing on backplane segments. CAMAC (IEEE 583, Fastbus's predecessor), used a branch-highway cable connecting crate controllers. The branch cable used a different protocol than the crate backplane.

Uniform addressing, so that each device can have as much address space as it needs, and with no dedicated positions on backplanes. Devices have an address which is unique in a connected system and which is independent of device position within a segment. All positions are equal in capability. Position-dependent addressing is also needed, at least during system initialization, for addressing a module to tell it its assigned position-independent address.

Sparse-data scanning capability, because the relevant data is usually a tiny fraction of the data available (most elements of a large detector are not hit by any particles at all in a typical event).

These goals, though partially conflicting, have been met to a remarkable degree by the Fastbus design.

## HARDWARE AND ORGANIZATION OF THE FASTBUS SYSTEM

The essential part of the Fastbus design is the bus protocol. Various electrical and mechanical implementations are possible which use this protocol and hence are easy to interface to one another, but only one electrical implementation and one module design have been chosen for standardization. However, module dimensions are specified so that the same module can be used in an air-cooled or in a conduction-cooled (water-cooled) crate.

The most common implementation consists of a crate which fits in a standard 19-inch rack, with a multilayer backplane at the rear and card guides at the top and bottom as shown in Figs. 1 and 2. Cooling air flows vertically from bottom to top through the crate, driven by separate fan units which may serve several stacked crates. Water cooled rechillers can be mounted between crates as necessary. The design capacity of 2000 watts per crate

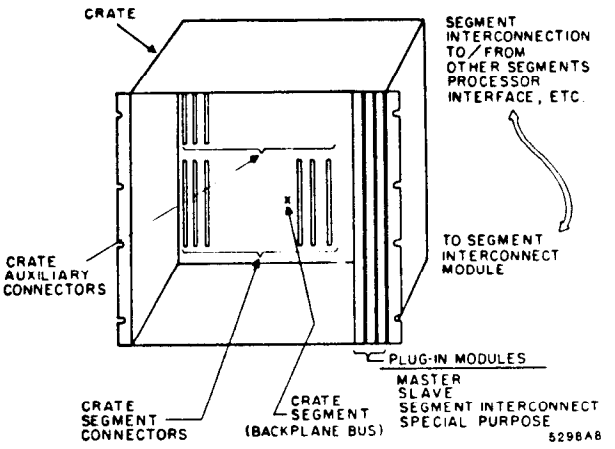


Fig. 1. Basic FASTBUS elements.

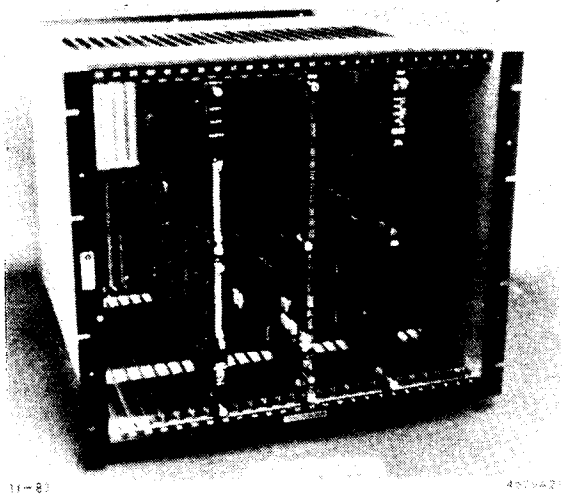

Fig. 2. FASTBUS crate and modules.

seems easily accommodated, and with care 3000 watts is possible.

A crate holds 26 modules (Figs. 3 and 4), which consist of circuit boards of approximately 366.7 mm (14.437 in.) high by 400.0 mm (15.748 in.) long, with optional front panels about 16 mm (0.63 in.) wide. The rear edge of the boards contains two box connectors which mate with square pins that protrude from the backplane. The main bus connector has two rows of 65 positions on 2.54 mm (0.1 in.) centers. An optional auxiliary connector has up to three rows of 65 positions, and connects to long non-bussed pins which pass through the backplane so that user cabling can be attached to the backplane rather than to the module. Small card guides are optionally provided on the rear of the crate to aid connector alignment and cable retention.

The main connector supplies power and ground as well as signals, with one ground to every four signals. Power

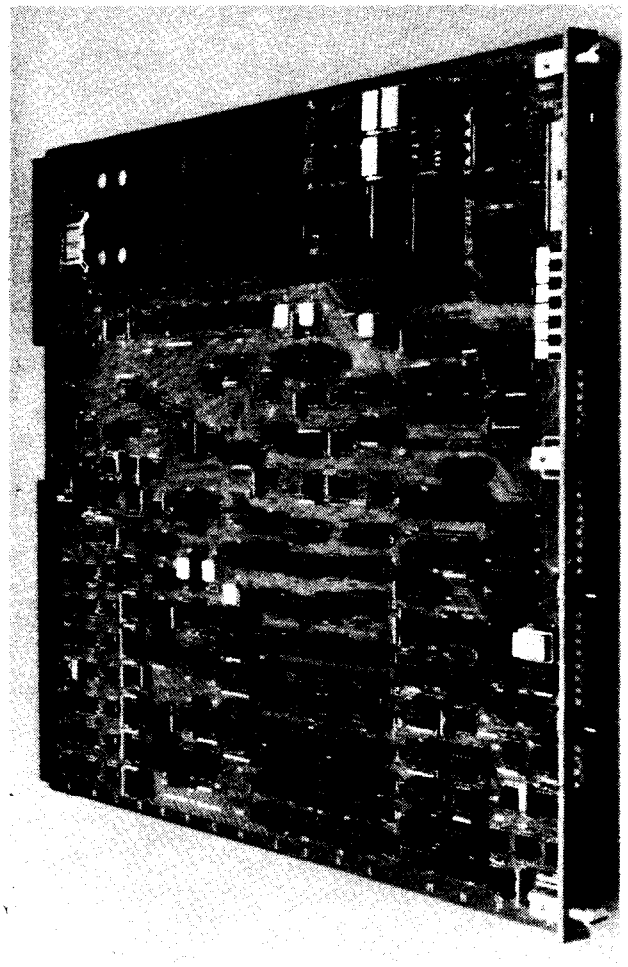

Fig. 3. The SLAC snoop module.

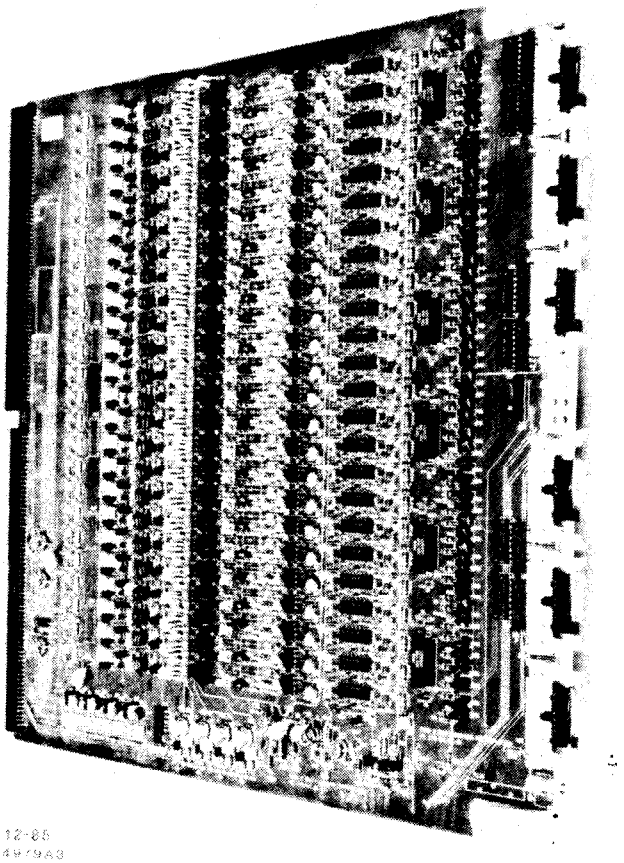

Fig. 4. SLAC Mark II/SLC drift chamber postamplifier module.

is distributed by heavy layers of copper in the backplane, providing for 300 amperes at +5 and −5.2 volts, 200 amperes at −2 volts, 50 amperes at +15 and −15 volts, and 100 amperes at +28 volts. A quiet analog ground line is also provided. Power supplies are separate from crates, connected by remote-sensing cables. Typically they will be mounted in the rear of the rack. The ±15 and +28-volt supplies are optional.

Bus terminators and a small amount of logic associated with bus arbitration, broadcast timing and geographical addressing reside on small boards mated to extended main-connector pins on the back of the backplane in the end positions. All bus signals are emitter-coupled-logic (ECL 10K) levels.

## SEGMENTATION OF THE FASTBUS

The backplane bus in a single crate is a single segment, i.e., corresponding pins in all backplane connectors are directly connected by printed circuit traces, which means that the bus can be driven by only one device at a time. Thus, although multiple processors may be plugged into the crate and share the single backplane bus, they must take turns using it. The mechanism that determines whose turn it is is called arbitration. Contention for the use of the bus may reduce the throughput of the system by causing processors to wait.

Distributing the processors among several crates reduces the contention problem if the data they need for operation is similarly distributed, but occasionally they will need access to one another's data. In a Fastbus system, there may be many segments that operate independently most of the time, but are temporarily linked together by segment interconnect modules when necessary for inter-segment data transfers.

The cables which interconnect backplane segments are segments themselves, and may contain devices other than segment interconnect modules. Devices on cable segments obey the Fastbus protocols, but must provide their own power because power is not included in the cables. The term "device" will often be used instead of "module", to emphasize the logical similarity of devices whether they fit in crates and attach to backplanes, or have arbitrary shapes and locations and attach to cable segments.

To transfer data, a device must first gain the right to use its bus segment via the arbitration mechanism, then assert onto the bus lines the address of the device it wants to communicate with. After communication is established, the address is removed and the bus lines are used for transferring data either to (reading) or from (writing) the originating device.

Segment interconnect modules monitor the activity on the two segments they connect, awaiting the appearance of any address in a set of addresses which they have been programmed to recognize. They respond to such an address by requesting use of their other segment and asserting the given address on that segment when they gain control. The two segments remain locked together until the operation is complete. An arbitrary number of segments can be successively linked as needed for a given operation. The address contains all the information needed to direct the appropriate segment interconnects to respond and form the correct connection.

In order to use the address to provide the routing information in a practical way, the total address space available to the system is divided among the segments in such a way that the most significant bits of the address are sufficient to specify which segment is addressed. Every device on a given segment then has the same value for the high-order part of its address, and that value can be thought of as the segment number. The less significant bits serve to specify which device on the given segment is addressed, while the least significant bits specify the part or function within the device which is addressed.

The segment interconnect modules can thus be implemented in a simple way by using the high-order address bits to address an internal memory which contains a "1" in the locations corresponding to addresses which are to be recognized and passed, and a "0" in the other locations. When the system is initialized, these memories are loaded with the patterns needed to route all operations correctly. These internal memories are called "route tables", though actual connection routes are only determined by the combination of all the route table memories in the system.

With this scheme, there are no restrictions on the kinds of interconnections which may be made between segments. Segments may be connected in a tree structure with a big computer at the trunk and data acquisition devices at the leaves, for example. If high traffic between two widely separated segments causes excessive interference with the intermediate segments, a cable segment can be added which bypasses the intermediate segments. No device address changes are required because of this change, and once the route tables in the segment interconnects are reinitialized to make use of the new route, the interfering traffic will disappear from the formerly intermediate segments. Tree, star and ring structures can all be accommodated. Figure 5 shows an example of a simple tree connection.

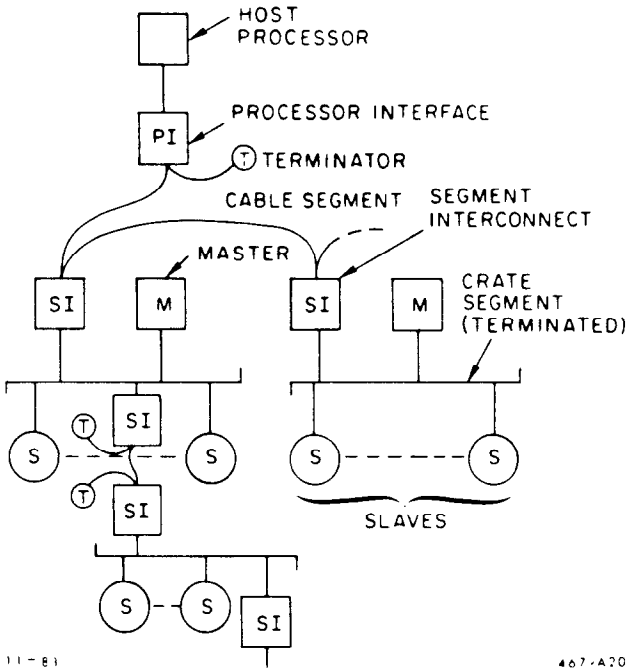


Fig. 5. Example of FASTBUS system topology.

However, some rules must be obeyed when setting up the route table information for the segment interconnects. For example, only one interconnect module may respond to any given address on a segment, since there must be only one path used for a given operation. A procedure has been developed for creating the routing information automatically, which makes it easy to reconfigure the system as needs change. Dedicated systems which do not change may avoid the need for route table initialization by storing the necessary route information in permanent read-only memories.

## THE FASTBUS PROTOCOL

The Fastbus uses the same 32-bit parallel bus (AD) for Addressing and for Data transfer, at different times. This technique is called time multiplexing. It may seem strange to multiplex address and data in a system which strives for ultimate speed, but it turns out that the speed penalty is not nearly as great as is usually supposed, because the data cannot be used until address recognition is complete. The reduction in the number of bus lines, connections and transceivers results in significant economic and reliability advantages for multiplexing. In order to achieve maximum speed for data transfer, block transfer modes have been developed in which a single address word is followed

by many data words, making the time penalty for multiplexing address and data insignificant. In fact, even if we were willing to implement 64 lines and transceivers we would probably choose to multiplex anyway, in order to get the greatest system throughput.

To initiate a transfer, a master device asserts the slave's address on the 32 AD lines followed by the address strobe, AS, as shown in Fig. 6. (A master is a device which initiates an operation by arbitrating to acquire control of the bus and then asserting an address. A slave is a device which responds to the address on the bus. A device may be able to act either as a master or as a slave at different times.) The address assertion sets up the path between master and slave, through segment interconnect modules if necessary. When the slave recognizes its own address, it responds with the address acknowledge signal AK. AS and AK remain asserted until the completion of the transfer, and serve to lock other users off the bus and cause them to ignore all bus activity. In fact, once the connection is established and the AS-AK lock is complete, the two devices could do almost anything with most other lines on the bus and no other devices would be disturbed. In order to facilitate the construction of compatible devices, however, standard protocols for most useful transfers have been specified.

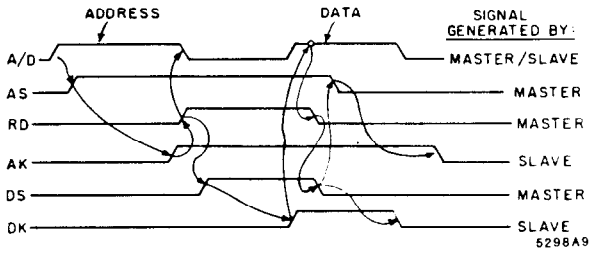


Fig. 6. Basic handshake read operation (as seen by master).

When the master sees the AK response, it knows that the address information is no longer needed, and removes it from the bus. It now asserts data and the data strobe DS, in case of a write operation, waiting for the acknowledging DK from the slave before removing the data. For a read operation, DS is asserted along with the read line RD, as shown in Fig. 6. The slave responds with data and DK. The transfer ends when the master sees DK and records the data on the AD lines, removes its signals including DS and AS, and the slave, seeing AS removed, removes its signals including AK. The connection between master and slave always has full handshaking at the beginning and end of a transfer.

The transfer of single data words requires two strobe edges, the assertion of DS and then its removal. In effect, the first edge controls the assertion of data and the second edge controls its removal. This restores the bus to an un-driven condition after each cycle, so that the direction of data flow can be reversed between two cycles, as shown in Fig. 8. Block transfers sacrifice the possibility of quick reversal and double the transfer rate by using both strobe edges to transfer data as shown in Fig. 7, without restoring the bus to its undriven condition until the very end of the block. Note that this is still a full handshake, so that the transfer cannot proceed without the agreement of both parties.

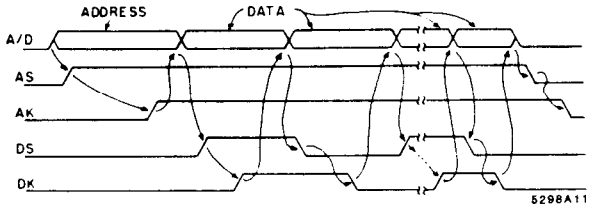


Fig. 7. Write block transfer (as seen by master).

However, it is also possible to perform a block transfer with pipelined handshake, between devices which can handle the same data rates. In the case of a write, the master simply asserts data words and DS transitions at whatever rate is appropriate, without waiting for the DK responses. In effect, DS becomes a clock which the slave uses to find the data words in a synchronous transmission. For a read, the master sends DS and the slave replies with data and DK; the master uses each DK transition to find the data words in the received stream of signals. Handshake-protected transfers require each data word to be on the bus for at least two bus-propagation delays, while the data flows to its destination and the acknowledge flows back to the source. On long cable segments, this delay can be significant, limiting system throughput. When pipelined handshake is used, however, several data words could possibly be flowing through the bus transmission lines at once. With pipelined handshake, data can be transmitted at the full bandwidth capacity of the medium.

In most cases, transfers will use full handshake protection. The handshake permits either party to pause for a moment if necessary (perhaps for refreshing dynamic memory chips), and allows either party to terminate an operation early (in case a buffer overflows, for example) with both parties having full knowledge of how many words were successfully transmitted. Transfers with pipelined handshake require the master to know the capabilities of the slave and the bandwidth of the entire path in order to choose a workable DS clock rate. If the transfer does not push this rate near its limit, one could

just as well have used full handshake protection and not had the worry.

The information that controls whether handshake or pipelined handshake is to be used and whether a block transfer (using both strobe edges) is to occur is encoded on two additional lines, the mode select (MS) lines. The MS code is also used with the address, to specify broadcast or normal addressing and to select normal data space or control register space access. Status information is supplied by the slave for each cycle, encoded on three SS lines, to inform the master of errors or unusual conditions. The MS lines act like extra AD lines which always carry address or data modifier information from master to slave, timed like address or write data. The SS lines always carry information from slave to master, timed like read data. Thus every cycle has timing compatible with both write and read, the only difference being which direction the information on the AD lines is flowing.

## ADDRESS-LOCKED AND ARBITRATION-LOCKED OPERATIONS

The transfers described above can be generalized to allow data flow to reverse direction without breaking the connection. For example, a read-modify-write operation as shown in Fig. 8 asserts address, reads data, turns the AD lines around again by removing the RD signal, and writes the modified data back to the slave. Such a transfer is uninterruptable by any other processor, since it is locked the whole time by the AS-AK lock and no other device can use the bus. It thus forms the kind of indivisible operation needed in multiple-processor systems for coordinating use of shared resources.

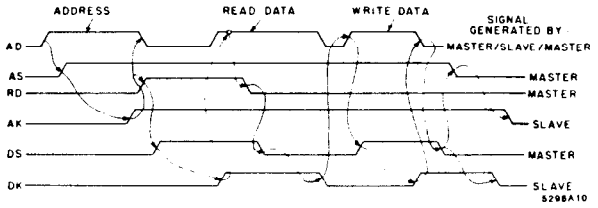


Fig. 8. Address locked operation: read-modify-write (as seen by master).

This idea can be extended to even more complex operations, so long as master and slave agree on the meaning of each bus cycle. For example, the address which connects master and slave could be followed by a data word which the slave interprets as an internal address or special command, followed by another data word which the slave interprets as data. One of the MS codes is assigned the meaning "internal address" to aid in this interpretation. Such a cycle is called a Secondary Address cycle, and is used frequently in the standard protocols, especially for accessing control and status registers.

Operations of this sort are referred to as address-locked operations. Note that the individual data words usually will not be sent as a block transfer, since the possibility of turning the bus around between words is being maintained, and the time needed for that is provided by the alternate edges of the data strobes. However, block transfers may also be included within address-locked operations.

A still more general kind of transfer on the bus is called an arbitration-locked operation sequence. It consists of a series of (possibly complicated) address-locked operations, between which the master does not release the bus for arbitration, so no other master can get control. This can be very useful for synchronizing a set of slave devices which are shared by several processors, so that a coherent set of operations can be performed without interference from other processors. This mechanism even works if the slave devices are on various different segments, because the segment interconnect is designed to maintain any connection until the originating master gives up its bus, even if the current operation is not passing through that segment interconnect. This behavior obviously reduces system throughput, but is essential for solving the multiprocessor multi-segment resource management problem efficiently. This behavior can be avoided when it is not needed, by merely allowing the master to release the bus after each access.

## GEOGRAPHICAL ADDRESSING

In systems of any significant size, the effort involved in correctly setting all module addresses by manual switches is too great. Furthermore, some means of automatically determining the locations of modules is needed, if only for use as a record or a check. When a system without switches is first turned on, the control registers which will contain the module address information are initialized randomly. Some other mechanism is needed to address the modules in order to load the device address register with the proper contents.

An addressing mechanism has been included in the Fastbus which allows accessing a module by its location in the system, or by its "geographical address" rather than by its normal or "logical address". The geographical address of a location on a segment is the position or slot number of the location. The geographical address of a segment is given by the high-order bits assigned to addresses on that segment (as determined by segment interconnect route table entries). Thus the geographical address of a particular module in the system is given by the appropriate high-order bits specifying the segment, many intermediate zero bits, and five low-order bits specifying the position or slot number on the segment. For economic reasons, addresses of this type are recognized by

a Geographic Address Controller packaged with the terminators at one end of the bus, which generates an EG (Enable Geographic) signal on the segment. Individual modules seeing EG compare the low-order address bits with the slot number as coded on five pins provided by the backplane and connector. It is permissible for modules to implement only this form of addressing, which makes them quite simple. Actually, the eight low-order address bits are used for Geographic Addressing, allowing for the combination of several physical crates into one logical crate, but only five bits are encoded in each backplane. On cable segments, no EG signal is provided and devices must decode the entire geographic address themselves. The cable "geographic" address must be set by switches in the device.

The geographical addressing mechanism can be used to access the control and information registers in uninitialized devices so that an automatic procedure can be used to initialize them.

## CONTROL AND STATUS REGISTER ADDRESSING

Certain registers and functions in devices need to be separated from the normal data registers in a way which provides some protection from accidental access and which does not interfere with the allocation of addresses to the normal data portions of the devices. For example, two memory devices should be able to have their addresses set so that the memories are adjacent in address space, allowing them to be used as one larger memory. However, they may contain control registers and status registers associated with logical address assignment, memory protection, or error detection and correction, and these registers must also be accessible in some way. Furthermore, it is desirable that devices have basic status and information registers in standard locations so that standard shared programs like the system initializer can find them easily.

The method chosen to accomplish this is a special case of the address-locked transfer. The device is selected by its address, with additional information on the MS lines to specify that this is really a control/status access. The first data word, called a "secondary address" cycle and labelled by a special MS code, is the number of the internal control/status register, and the second is the data to be read or written. This three-cycle transfer provides a full 32-bit address for use within a device, which is enough address space so that it can easily be allocated in standard ways without fear of a shortage. At least the more complex devices should include a read-only memory containing information about the device and its properties, which can be used by programs and people to make managing large systems easier. A large block of addresses within the device has been reserved for this purpose, and a file and directory structure has been defined. Standard locations have also been specified for all the usual control and status bits. One register is mandatory in all devices, containing a device identification code and several other useful bits.

With this scheme, the Control and Status Registers (CSRs) of any device in the system are easy to find. Either the device's geographical address or any logical address that it responds to may be used to establish contact, whereupon the desired register can be accessed via a secondary address cycle.

## BROADCAST OPERATIONS

A broadcast operation is one in which a single master sends information to multiple slaves. Broadcasts can be used to synchronize devices or clear a bank of counters. Since more than one slave may be involved, no handshake between slave and master is possible. However, a system handshake has been devised which informs the master that his command has propagated to every segment to which it was addressed. The master asserts an address with a code on the MS lines indicating that a broadcast is to occur. The address may refer to a specific segment or may refer to all connected segments in a pattern controlled by the route tables in the segment interconnects, or it may specify all segments beyond a given segment in that predefined pattern.

The general address used for broadcasting has zeroes in its most significant bit positions, so that the route table entries corresponding to the zero address are used for routing broadcasts. For this entry, more than one segment interconnect may recognize the address and pass it onward, since no handshakes are to be returned. The pattern formed by the pathways propagating from the broadcast master must form a simple tree with no cross connections, another rule to be applied by the initialization program.

When the broadcast address has propagated successfully throughout the system, the system handshake occurs and the master asserts the control register number to which it wishes to broadcast, following the protocol of the control/status register addressing discussed above. When the system handshake is returned, the control data is asserted. Thus, any kind of standard control operation may be performed at once on a large set of devices by a broadcast. Broadcasts may also be made to ordinary data space, using a secondary-address data cycle to select the appropriate internal data location in the selected devices (the address used for broadcast has no room for this information, and each device is at a different address anyway). Data space broadcasts are rarely useful unless one of the selective broadcast modes is used, or the scope

of the broadcast is limited to a part of the system containing many identical devices. Broadcast read (broadcall) is also permitted, but is only useful under special conditions because it results in the logical OR of the data provided by every selected device.

Broadcasts may take some time to start, since they must wait for completion of all conflicting use of the segments involved. Applications requiring very fast response to signals from a central controller may have to resort to direct cables, since there is no way to achieve fast response in a reliable way in a multiple-segment system. Once the system connection is complete, however, speed of execution of the data cycles is limited only by signal propagation delays, so a reasonably synchronous execution of the command is achieved. At least, one can be certain that all devices will see the command before any other bus operations will occur.

## SPARSE DATA SCANS AND THE "T" PIN

The Fastbus design includes a pin called the "T" pin, which connects inside the backplane to the AD bus line corresponding to the module position number. Thus, a module in position 12 finds its T pin connected to AD12, etc. The T pin thus can be used for positional information.

The T pin was originally included in the Fastbus to provide a means for rapidly scanning sparse data in detector front-end modules. A controller on the backplane broadcasts a command to the front-end modules which causes them to assert their T pins if they contain data. The resulting pattern on the AD lines shows the controller immediately which modules need to be read out, thus avoiding the overhead of polling them one at a time. Other useful broadcast operations using the T pin have been defined, for discovering which crate positions are occupied or which devices are asserting the service request line. The T-pin is simulated on cable segments.

## INTERRUPTS ON THE FASTBUS

An interrupt is a request from some device to some processor for service or attention. Since interrupts may have to cross segment boundaries, and since they must carry information, they are handled by normal Fastbus operations. The interrupting device addresses an interrupt-sensing control register in some processor, and writes its own address and possibly other information into the register. The processor then has all the information it needs to find the interrupting device and service it at some later time.

In some systems, large numbers of simple devices may need to signal a request for service without having the capability of gaining bus mastership and performing an interrupt write. Within a single segment, such devices may assert a service request (SR) line, which can be monitored by a special interrupt service device which does have the circuitry to gain mastership and find the requester, whether by means of the T pin or by polling or some other means. The interrupt service device may then perform the necessary service itself, or it may send a normal interrupt message on behalf of the simple requester to some other processor. The SR lines may be passed through selected segment interconnect modules as well, allowing multiple-crate extensions of the simple SR system where appropriate.

## ARBITRATION FOR FASTBUS MASTERSHIP

Since multiple devices on a segment may wish to become master of the segment, some means is needed to prevent more than one of them from using the bus at a time. Ten lines on the bus are dedicated to the solution of this problem. Six of the lines are used to hold a "priority" code that determines which competing device wins mastership, while the other four are used to synchronize the requests. The arbitration mechanism operates in parallel with use of the bus, so that little time need be wasted in switching from one master to another.

At a given time, each requester tries to assert its priority on the AL (arbitration level) lines. The lines perform a "wire-OR" function, so that any asserting requester overrides nonasserting requesters at each bit position. Each requester compares his level with the level on the AL lines bit by bit, from most to least significant. If it sees an AL line asserted which it did not assert, it removes its assertions of all less significant bits, because it knows that a higher priority requester is competing. After four bus propagation delays, only the highest arbitration level remains asserted and each requester knows whether it has won or lost.

Of the 64 possible levels, zero is not used because it is easily confused with an idle bus, 1 through 31 are available for use within the segment, and 32 through 63 are used as "super" priorities which must be assigned uniquely throughout an entire connected system. The normal levels 1-31 must be assigned uniquely to devices within a given segment, but exist on every segment to be used over and over again. When a segment interconnect connects a master to another segment, the arbitration level used on the second segment will normally be the local level of the segment interconnect module rather than that of the originating master. However, if one of the super priorities was used by the master, the segment interconnect will propagate that level onto the second segment, which it is free to do since the super priorities are unique within the system. The super priorities can be useful in preventing undue delay for important broadcasts, and can help

expedite important messages, which otherwise may suffer from fluctuating levels as they form paths through the system.

The current master determines when it will be finished with the bus, and releases the arbitration circuitry so that the next master can be selected before it finishes. It thus maintains ownership of the bus as long as it likes, which is the mechanism used to implement the arbitration-locked operations discussed above.

An "Assured-Access" protocol is also available, which provides a kind of round-robin access to the bus, avoiding bus-hogging by high-priority devices. Assured Access works by preventing a master from reapplying for mastership once he has had it, until no other applicants remain. Most devices should use Assured Access in normal operation of a system, but the choice is up to the system configurer. Priority access can be mixed with Assured Access as appropriate—the priority devices simply apply for mastership and join the arbitration process whenever they wish.

The term "priority" is somewhat misleading, because there is no mechanism to allow a high-priority device to preempt or force a lesser one off the bus. In a lightly loaded system, first-come first-served is the dominant mode of behavior. The arbitration level only serves to break ties when simultaneous requests for use of the bus occur, or when requests become synchronized as a result of waiting for passing traffic. Thus arbitration priority should not be confused with the kind of priority which may apply to multitasking executive programs or computer interrupt systems which allow nesting.

## DEADLOCK PREVENTION IN MULTI-PROCESSOR SYSTEMS

Deadlock is a fundamental problem which must be solved in multi-processor systems, caused by conflicting exclusive-access requirements of processors for multiple resources. Fastbus provides some tools which help to solve this problem. For example, address-locked operations allow reliably testing and setting semaphores without interference by other processors. A "User Address Register" allows administration of resource ownership in a distributed, cooperative system without requiring a central software resource manager. Especially important in large systems, Segment Interconnects hold any paths which they establish, until the master which originated the connection gives up the bus to another master. This allows one master to block access by any other to critical resources located on several other segments, to gather all the needed resources and protect them before it begins taking any irreversible actions—if it cannot get access to all the required resources, it releases the bus and tries again later. Ultimately, when avoidance fails, Fastbus relies on a time-out to resolve deadlocks: never wait forever for anything; give up, wait a random time and try again.

## DIAGNOSTIC NETWORK

Diagnosing problems in a complex system with multiple bus segments requires powerful tools to set up tests and gather information from multiple sources, then bring it together for analysis and display. Because the Fastbus interconnect system might be the point of failure, and because it is desirable to be able to collect information about the system without disturbing the system, a secondary information path is needed.

Fastbus has allocated two lines in the backplane for use by a diagnostic serial network. The protocols have not yet been standardized, but the principles are clear. The network must be robust, easy to connect, available everywhere in the system, inexpensive, versatile. Two modules on the same backplane should be able to communicate just as well as modules that are widely separated. This can be achieved by using an Ethernet-like scheme, with every module transmitting on the TX serial line while listening to the RX serial line. The TX line is a normal wire-OR backplane signal line, which is received by a network interface attached to the back of the backplane. (Eventually this should be part of the terminator/ancillary logic board.) The interface drives a network coaxial cable via isolating transformers, and also receives cable signals and drives the RX line with them. Thus the TX signal from any module on any backplane is visible at every module position in the system, on the RX line. The network cable must visit cable-segment devices individually as needed.

To make the cabling convenient, to reduce restrictions on cable flexibility, length, and quality, and to reduce board-space requirements in devices that wish to use the network, the data rate must be rather modest.

SLAC is presently working on an implementation of the AppleTalk (TM) network, used primarily by the Apple Macintosh and LaserWriter, as a prototype diagnostic network. It operates at 230400 bits/second, and needs only half of a Zilog SCC 8530 communications chip in each device. AppleTalk has about the right combination of attributes for this application.

The prototype implementation is being done in the SLAC Fastbus Snoop Module,[2] which is a sort of specialized fast logic analyzer which understands Fastbus protocols, can store a history of bus activity, set traps and triggers, and act as a master to exercise remote parts of the system for test purposes.

## CABLE SEGMENTS

The Fastbus cable segment contains all the protocol and data lines needed for full Fastbus operation. It does not carry power, daisy-chains, T pin, geographical address encoding pins, serial network lines or free-use lines. The geographical address encoding must be provided by switches and the T-pin connection must be simulated in the devices which attach to the cable segment. Otherwise, devices connected to cable segments act just like devices (modules) connected to backplane segments.

The wire-OR behavior of certain Fastbus lines (especially the arbitration lines) is fundamental to the protocol. Wire-OR in the usual style (as used on the backplane segments) has certain unavoidable limitations[3] caused by the use of voltage-driver technology. On the backplane, circuit delays are used to overcome these problems, but on long cable segments this solution would cause unacceptable delays.

Therefore, a new transceiver technology was developed for use on Fastbus cable segments, which uses current drivers and voltage receivers to completely eliminate wire-OR problems. The signals from multiple drivers simply add, the laws of linear superposition apply, and the receivers are comparators which only need to discriminate between the "0" and "1-or-more" logic levels. In addition, long cables are especially vulnerable to electrical noise, ground potential differences etc., so the cable segment uses differential signalling to cancel out these effects. The resulting system behaves in a nearly ideal way. Present implementations use hybrid technology for the transceivers, but monolithic technology is already capable of the required performance so fully integrated transceivers should eventually be available.

## CONCLUSION

The Fastbus design evolved over a seven-year period into a simple, clean and cost-effective system which can solve a broad spectrum of problems. Its ability to cope with extremely fast as well as slow devices, its easy expandability, its parallelism, modularity, and multiprocessor support commend it for a wide range of applications.

Recent experience at CERN involving bids on large systems (private communication from Henk Verweij) reveals that Fastbus has already begun to show its expected economic advantage. The cost of usable module area, after subtracting system interface overheads etc., for a complete system including power and cooling, was found to be: SFr 0.33/cm$^2$ for Fastbus, SFr 0.55/cm$^2$ for CAMAC, and SFr 0.75/cm$^2$ for VME. Most people consider VME to be very inexpensive, so this result is quite impressive. It should get even better when Fastbus designs begin using the LSI gate-array interface chips which have just become available.

## FOR MORE INFORMATION

The status of the Fastbus is reported annually at the IEEE Nuclear Science Symposium. The Symposium proceedings are published as the IEEE Transactions on Nuclear Science each February. For example, see the articles by H. V. Walz and E. J. Barsotti,[4] H. Verweij[5] and others.

For current information, contact Louis Costrell, Chairman, U. S. NIM Committee, National Bureau of Standards, Center for Radiation Research, Washington, D.C. 20234, telephone (301) 921-2518.

Several useful articles about buses appear in the August 1984 issue of IEEE Micro. There are detailed discussions of bus signal propagation,[6] arbitration[7] and protocol.[8] These describe the IEEE P896 Futurebus in particular, but that bus has some similarity to Fastbus, and the physics problems are the same. Note that the Fastbus solution to the bus driving problem was the use of ECL 10K transceivers, which behave almost exactly like the new transceivers described in Ref. 6, except that they operate at different voltage levels. Fastbus arbitration behaves just like the system described in Ref. 7 except that the control mechanism is different. Signal diagrams for other buses are usually upside-down compared to Fastbus, because ECL 10K signals perform the wire-OR going positive while most other systems wire-OR going negative—i.e., any transmitter being active pulls the signal low. The arbitration article also contains a good discussion of the wire-OR problem. There is also an elementary tutorial on buses in that issue.[9]

## ACKNOWLEDGMENTS

This is a major revision of an earlier paper.[10] Several figures were taken from the Fastbus specification,[1] which in turn adopted some text from my earlier paper, which explains certain similarities that the reader may notice. The Fastbus specification was very ably edited by Ken Dawson, now of TRIUMF, and contains a great deal of helpful tutorial material.

Helmut Walz and Louis Costrell were helpful in providing figures for use here. I am also grateful to Jerry Friedman and Ray Larsen for encouragement and support, and to Johannes Joemann and David Gelphman for their assistance with the AppleTalk project.

## REFERENCES

1. ANSI/IEEE std 960-1985, FASTBUS Modular High Speed Data Acquisition and Control System.
2. David B. Gustavson and Helmut V. Walz, "SLAC FASTBUS SNOOP MODULE—TEST RESULTS AND SUPPORT SOFTWARE", to appear in IEEE Transactions on Nuclear Science, February 1986.

3. D. B. Gustavson and John Theus, "Wire-OR Logic on Transmission Lines", IEEE Micro, Vol. 3, No. 3, June 1983, pp. 51-55.

4. H. V. Walz and E. J. Barsotti, "FASTBUS REVIEW 1985", to appear in IEEE Trans. Nucl. Sci., February 1986.

5. H. Verweij, "FASTBUS IN EXPERIMENTS IN EUROPE", to appear in IEEE Trans. Nucl. Sci., February 1986.

6. R. V. Balakrishnan, "The Proposed IEEE 896 Futurebus—A Solution to the Bus Driving Problem", IEEE Micro, Vol. 4, No. 4, August 1984, pp. 23-27.

7. D. M. Taub, "Arbitration and Control Acquisition in the Proposed IEEE 896 Futurebus", IEEE Micro, Vol. 4, No. 4, August 1984, pp. 28-41.

8. Paul Borrill and John Theus, "An Advanced Communication Protocol for the Proposed IEEE 896 Futurebus", IEEE Micro, Vol. 4, No. 4, August 1984, pp. 42-56.

9. David B. Gustavson, "Computer Buses—A Tutorial", IEEE Micro, Vol. 4, No. 4, August 1984, pp. 7-22.

10. D. B. Gustavson, "An Introduction to the FASTBUS", Nucl. Phys. A335(1980), p. 571-578 (superceded by this paper).

## BIOGRAPHICAL SKETCH

David B. Gustavson has been a member of the Computation Research Group at the Stanford Linear Accelerator Center in Palo Alto, California, since 1977. He is currently working on a diagnostic system, human interface and local network for the Fastbus. He also serves as chairman of the Fastbus Software Working Group and is a member of the hardware design team and of the executive committee.

A member of the IEEE, IEEE Computer Society, and ACM, Gustavson received a BS in physics and mathematics from the University of Nebraska in 1962, studied at the Georg August Universität in Göttingen, West Germany, as a Fulbright Fellow in 1962-1963, and received a PhD in high-energy elementary particle physics from Stanford University in 1969.