



Fermi National Accelerator Laboratory

FERMILAB-Conf-90/61

Effects of Various Event Building Techniques on Data Acquisition System Architectures*

Ed Barsotti, Alexander Booth and Mark Bowden
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, Illinois 60510

April 1990

* Invited talk presented at the 1990 Conference on Computing in High Energy Physics, Santa Fe, New Mexico, April 9-13, 1990.



Operated by Universities Research Association Inc. under contract with the United States Department of Energy

**EFFECTS OF VARIOUS EVENT BUILDING TECHNIQUES
ON DATA ACQUISITION SYSTEM ARCHITECTURES**

Ed Barsotti, Alexander Booth & Mark Bowden
Fermilab
P.O. Box 500, Batavia, IL 60510

April 1990

TABLE OF CONTENTS

ABSTRACT.....	1
INTRODUCTION.....	1
EVENT BUILDING OVERVIEW.....	2
EVENT BUILDING & INTERCONNECTION NETWORKS.....	2
Interconnection Network Terminology.....	3
Time-Shared Bus Architectures.....	4
Multiple Bus Interconnection Networks.....	6
Multiport Memory.....	8
Crossbar and Other Switch-Based Interconnection Networks.....	9
Integrated Processor Interconnection Networks.....	13
Communication Networks Applied to Event Building.....	13
ARCHITECTURAL CONSIDERATIONS.....	15
ADDING FAULT TOLERANCE TO INTERCONNECTION NETWORKS.....	16
SELF-ROUTING TECHNIQUES.....	17
INPUT AND OUTPUT QUEUEING IN AN INTERCONNECTION NETWORK.....	19
A FUTURE DATA ACQUISITION SYSTEM ARCHITECTURE & SOME FUTURE TECHNOLOGIES.....	20
Front-End Architecture.....	22
Pre-Processing.....	23
Data Links.....	24
Parallel Event Builders.....	25
Online Processor Arrays.....	25
FERMILAB'S DATA ACQUISITION SYSTEM ARCHITECTURE & PARALLEL EVENT BUILDER PROTOTYPE PROJECT.....	26
Switch-Based Parallel Event Builder Operation.....	27
N-Input & M-Output Barrel Shift Interconnection Network.....	29
Input & Output Time Slot Interchangers.....	29
Switch-Based Parallel Event Builder Integration.....	29
Control Modes Of Operation.....	31
Behavioral Modeling & Simulations Of The Architecture.....	32
System Design Methodology.....	33
SUMMARY.....	34
ACKNOWLEDGEMENTS.....	35
REFERENCES.....	35

EFFECTS OF VARIOUS EVENT BUILDING TECHNIQUES ON DATA ACQUISITION SYSTEM ARCHITECTURES

Ed Barsotti, Alexander Booth & Mark Bowden
Fermilab*
P.O. Box 500, Batavia, IL 60510

ABSTRACT

The preliminary specifications for various new detectors throughout the world including those at the Superconducting Super Collider (SSC) already make it clear that existing event building techniques will be inadequate for the high trigger and data rates anticipated for these detectors. In the world of high-energy physics many approaches have been taken to solving the problem of reading out data from a whole detector and presenting a complete event to the physicist, while simultaneously keeping deadtime to a minimum. This paper includes a review of multiprocessor and telecommunications interconnection networks and how these networks relate to event building in general, illustrating advantages and disadvantages of the various approaches. It presents a more detailed study of recent research into new event building techniques which incorporate much greater parallelism to better accommodate high data rates. The future in areas such as front-end electronics architectures, high-speed data links, event building and online processor arrays is also examined. Finally, details of a scalable parallel data acquisition system architecture being developed at Fermilab are given.

INTRODUCTION

The demands on data acquisition systems for high-energy physics experiments are increasing at a rapid rate due to the higher luminosities and interaction rates. From the early days of high-energy physics to most present-day experiments, when readout of a physics event is initiated, triggering on subsequent events is disabled until readout is complete. Other factors in an experiment contribute to this experiment "deadtime" but readout time is the dominating factor. Typically "deadtime", measured as a percentage, is held to less than 10% and is approximately equal to the ratio between event readout time and trigger rate times 100%. Now, with very high interaction rates and consequently very high trigger rates, readout time is an even larger fraction of the time between triggers.

New techniques for physics event readout ("event building") are now essential if we are to minimize deadtime. Several events worth of data must be buffered on or near the detector during triggering, such that when an acceptable trigger occurs, the buffered data for that event may be readout quickly, and without disabling the trigger.

Event builders, the devices used to readout event data, have evolved from simple single channel 'funnels' through a minicomputer bus, to multiple parallel channels (each with their own 'funnel' or bottleneck characteristics) feeding arrays (farms) of processors. More and more experiments are implementing event builders with increased parallelism for higher throughput. When one considers particle beam crossing times of 16 nanoseconds and subsequent very high trigger rates, it is clear that SSC detector data acquisition systems will require the use of totally parallel event builders with no inherent bottlenecks in addition to much larger amounts of pre-event builder buffering in order to achieve minimal deadtime experiments.

* Operated by Universities Research Association under contract to the U.S. Department of Energy

This paper will give a brief overview of event building in general, followed by a more detailed study of recent event building techniques and how these techniques affect data acquisition system architectures and performance. Analogies between various multiprocessor interconnection and telecommunications switching networks such as shared busses, crossbars, hypercubes, etc. and event building techniques will be detailed. Performance issues such as timing (synchronous or asynchronous), control (centralized or decentralized, data driven or data read), coupling (tight or loose) and buffering methods (store and forward or direct link) will also be detailed. Future techniques for event building such as opto-electronic or totally optical interconnection networks will be discussed. The paper will conclude with a brief look at future system requirements and a proposed new data acquisition system architecture being developed at Fermilab.

EVENT BUILDING OVERVIEW

Only a small fraction of the total data from each event is available for use in the initial trigger decision. The remaining data is scattered over many front-end buffers and must be collected in one place for detailed analysis. An "event builder" is the device in a data acquisition system which provides a physical connection between the individual data sources (detector front-end electronics) and the data destinations (high-level event processors or online data storage).

Regardless of the implementation, all event builders function as simple data multiplexers. If data rates are low, this multiplexing operation can take place over a single time-shared bus using software controlled selection of source and destination. This is the technique used in the majority of data acquisition systems to date. High-speed event builders have not been necessary because the data rates which could be supported by the sources and destinations were limited. This situation is changing rapidly. The ability to acquire, digitize and buffer data using VLSI front-end circuitry has increased allowable trigger rates by a factor of at least 1000. Similarly, the performance of high-level processors and the density of on-line data storage have both improved by a factor of almost 1000 over the last fifteen years. Unfortunately, the speeds of standard busses used for event building have improved by only a factor of ten in the same time period. The event builder has become the bottleneck.

There are two possible solutions to this problem. Either the trigger efficiency can be increased, limiting data rates to the bandwidth of the event builder, or the event builder bandwidth can be increased. Techniques for improving trigger efficiency are dependent on the experiment. Techniques for improving event builder bandwidth can be considered independently, as in the following comparisons.

EVENT BUILDING & INTERCONNECTION NETWORKS

Figure 1 shows a generic Interconnection Network (IN) used in multiprocessor and telecommunications systems. In high energy physics, the data source (S) is typically a detector subsystem and the destination (D) is a programmable processor. The IN and its associated control is referred to as an "event builder". Because the pattern of data flow is well defined (unidirectional and evenly distributed), a general-purpose IN can often be simplified for use as an event builder. An enormous advantage is gained for high energy physics if we are able to draw from both the computer and telecommunications industries when designing data acquisition systems requiring parallel event builders.

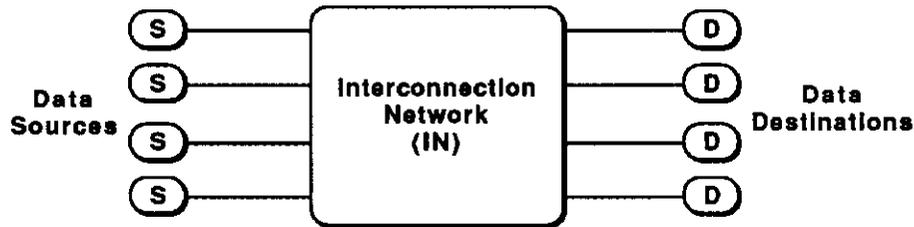


Figure 1: Generic Interconnection Network

Interconnection Network Terminology

To better understand how both multiprocessor and telephone switching system interconnection network theory and technology can be applied to event building in high energy physics experiments, the following explains classifications of various interconnection networks [1]. An Interconnection Network is a set of busses, switches and/or data links that permit connection between two or more devices. In a multiprocessor environment, processors are usually connected to memory components and other processors. The method of interconnection can be classified by three distinct characteristics; timing, transfer and control mode. The effect of each of these characteristics on IN performance will be discussed in individual IN architecture subsections.

A network is either "blocking" or "non-blocking". Blocking occurs when information cannot be transmitted through the network due to competition for the same internal or external datapath. "Output" blocking occurs when two sources attempt to simultaneously transmit to the same destination. "Internal" blocking occurs when two sources are transmitting to different destinations (or the same destination), but the messages must cross the same internal node of the network. For effective use as a parallel event builder, a network should have little or no blocking. Many networks which are inherently blocking can be made non-blocking by correctly time-ordering or distributing data which enters the network. This is difficult in a general-purpose network with random traffic, but is much less difficult in event building where the connection patterns are well defined.

There are two types of timing modes in an IN, synchronous and asynchronous. In a synchronous IN, a global or master clock exists and is used to lock-step actions within the IN. Asynchronous INs operate without a global clock. Communications occur via interlocked hand shaking. Asynchronous INs are more easily expandable and have the potential for higher throughput than synchronous INs but are more difficult to build and maintain.

There are two types of message transfer modes in an IN, packet-switched and circuit-switched. In a packet-switched IN, messages are broken up into smaller "packets" which are transmitted through a network in a "store and forward" mode. No complete link through the network is made prior to transmission of the first packet. The only requirement is that the next stage in the IN be ready to receive a packet. When a stage has received or "stored" a packet and a succeeding stage is ready to receive a packet, the first stage transmits or "forwards" the packet to this next IN stage. This action occurs until the message has reached its final destination. Most packet switching INs are "self routing" in that there is no pre-selected path for the packet. Its "route" depends on header words in the packet. In a circuit-switched IN, a complete physical

path from source to destination is established before the message is transmitted. Circuit-switched INs are usually more suitable for long messages whereas packet-switched INs are usually more suitable for short messages. Combinations of these two techniques are possible (circuit switching of data packets).

There are also two modes for controlling the flow of messages through an IN, centralized and decentralized control. All message flow control signals originate from a single source in a centralized control IN, quite often creating a funnel or bottleneck to message flow and adversely affecting the performance of the IN. This source of control must necessarily be complex to allow good system performance and still maintain centralized control. The understanding of the IN and its maintenance are usually simplified when centralized control is implemented. In a decentralized control IN, each component performs its own control. Multistage interconnection networks (MINs) are almost always decentralized control INs and are quite often also self-routing INs. Crossbar switch INs are typically centralized control INs. Multiple bus INs can be either centralized or decentralized INs.

Defining an IN by timing, transfer and control modes leads to eight possible classifications of networks. For example, a CSD interconnection network establishes a link from source to destination and then transmits the entire message (circuit-switched), operates with a global clock (synchronous) and has no central message flow control component (decentralized).

Time-Shared Bus Architectures

The shared bus, shown in Figure 2, is the most common method of interconnecting multiple sources and destinations. Bus bandwidths of several tens of Megabytes/second can be supported during block transfers, but the average data rate is usually much less due to the overhead of processor setup and bus access protocols.

A single shared bus has the advantages of simple control and low cost. It also provides bidirectional transfer capability for download and initialization. With repeaters it can scale indefinitely, although the total bandwidth does not increase and will usually decrease. The main cost element is the need for high-speed interface circuitry, which must be designed to support the full transfer rate of the bus even if each module is connected for only a small fraction of the total readout time. Failure of the bus itself will disable the entire system, but failure of an individual module is usually not critical.

In most cases, data readout is controlled entirely by the processors. A processor will arbitrate for the bus and then read event data from each of the front-end buffers before releasing the bus to the next ready processor. In more complicated systems, an intermediate event builder will read the front-end buffers and then write data directly into the memory of a selected processor. In some architectures several independent busses may operate in a parallel tree structure to reduce deadtime at the front-end. However, without intermediate data compression, the net bandwidth in a tree structured system is always equal to that of a single bus.

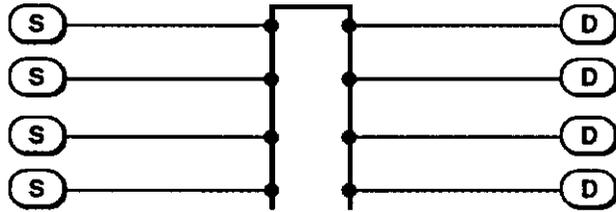


Figure 2: Shared Bus Interconnection Network

Great effort has gone into the development of both general-purpose and specialized busses for high energy physics applications. CAMAC [2], with bus transfer rates up to three Megabytes/second, has been in use since the early 1970s. FASTBUS [3], with bus transfer rates up to fifty Megabytes/second, has been in use since 1979. The industry standard VMEbus [4], with bus transfer rates up to thirty Megabytes/second, is also found in many systems, but mainly at the higher levels where commercial modules are available. Designers of the next generation Futurebus expect to exceed one Gigabyte/second with a very wide bus and self-timed data transfers. This is not likely to be realized in practice.

In commercial multiprocessor systems, most of the "low-end" machines (Silicon Graphics [5], Solbourne [6], etc.) use a shared bus architecture. Local cache memory on each processor module limits the need for continuous bus activity. The same effect applies to data acquisition systems where the time to read out an event is usually much shorter than the processing time.

One example of a shared bus system in high-energy physics is the event building and online processor farm sections of the data acquisition system for the Collider Detector at Fermilab (CDF) [7], as shown in Figure 3. The Event Builder reads out the front-end scanners, which have already read out ADC and TDC data, over two FASTBUS Cable Segments. After reformatting, the data is written over a single FASTBUS backplane to a VMEbus interface to a Level 3 online processor farm. Once a processor in the farm has processed the event by applying some filtering algorithm, it sets an attention flag which is read by a VAX [8] computer via FASTBUS and VMEbus. If the event is to be analyzed online, it is read out of the processor memory over VMEbus, then over FASTBUS and into one of the Consumer VAXs.

One of the inherent bottlenecks of the CDF data acquisition system is the mixture of data and control over the same busses. The FASTBUS network is shared between many devices, some sending control messages to initiate readout of the "next" event from the front-end electronics, others reading and writing data, others polling devices to see if they have completed a set task, etc.

The VMEbus in the Level 3 online processor farm is also shared by devices which write data, read data, and poll processors. This sharing is facilitated in both FASTBUS and VMEbus by arbitration mechanisms, but this sharing does bring with it significant implications in terms of reduced bandwidth and bottlenecks.

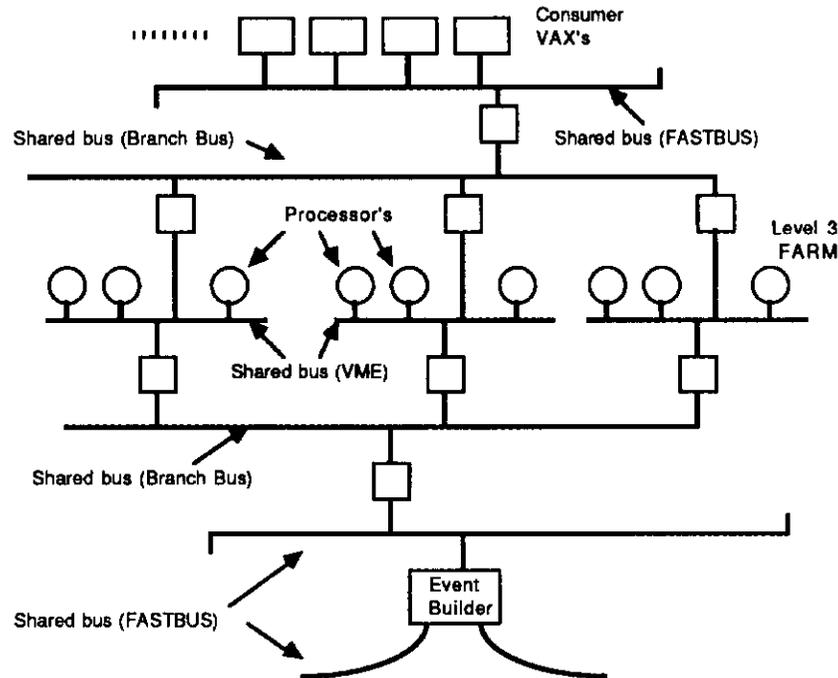


Figure 3: CDF Data Acquisition System

The inherent bandwidth of the bus is not a limiting factor for many existing systems. Instead, the interfaces and buffers at the source and destination modules create much of the bottleneck. Recognizing that the cost and error rate of electronics increases exponentially with the operating speed, a better solution to the bandwidth limitation is parallelism. Parallelism relies on multiple, lower-speed connections or components in place of a single, less-reliable, high-speed path. Parallelism does not necessarily imply more hardware. A system built using many low performance components will often cost less and occupy less space than a single, complicated high-performance device.

Multiple Bus Interconnection Networks

Many standard bus specifications and multiprocessor implementations define a second or third bus (Figure 4) which can operate in parallel with the main system bus. Additional bandwidth is gained only if processors do not contend for the same global resources. Examples include the VSB bus in VMEbus systems and the iLBX bus in Multibus [9] systems. This approach is usually limited to one or two additional busses by the physical packaging constraints of standardized systems. A multiple bus architecture can be very reliable since failure of any single bus has no adverse effect other than a reduction in total system bandwidth.

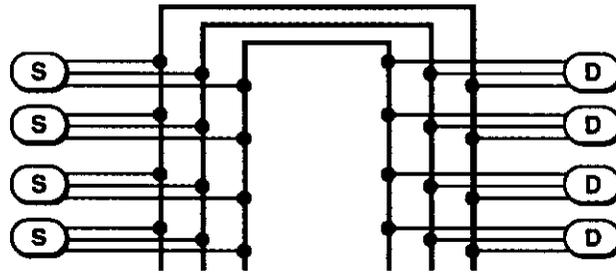


Figure 4: Multiple Bus Interconnection Network

With multiple busses, several events can be read out simultaneously. If events are assigned to specific buffers, then simple bus arbitration can be used to control readout sequences. Otherwise, a small amount of centralized control is necessary. As in any parallel system, the front-end buffers must be able to hold more than one complete event.

An example of a Multiple Bus Interconnect is the Heidelberg/Darmstadt Crystal Ball detector data acquisition system [10] shown in Figure 5 and consisting of FASTBUS and CAMAC front-end electronics, the Heidelberg POLYP multiprocessor system and an online VAX computer. The POLYP multiprocessor system consists of thirty Motorola 68000 microprocessors which are used to process the event data stream. The processors have their own local bus which is connected by bus switches to a the global POLYBUS. The data flows from the detector through the FASTBUS and CAMAC front-end electronics into a set of POLYP input processors which also buffer the event data. From here the data is transferred to the POLYP online filter processor's over the POLYBUS. Events which pass the filtering stage are again transferred over the POLYBUS to a POLYP I/O processor, where the data is read by the host interface and written to tape.

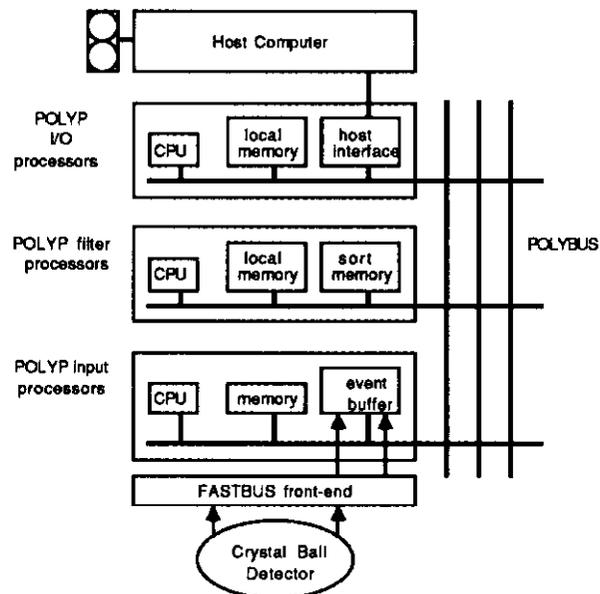


Figure 5: Heidelberg/Darmstadt Crystal Ball Data Acquisition System

Multiport Memory

Both the data sources and destinations in the multiple bus architecture are multiported, but the same bandwidth can be obtained with multiple ports on only one side of the interconnect as shown in Figure 6. Reliability is reduced because there is only one path from a particular source to a particular destination. Arbitration is handled by the multiport module rather than the bus.

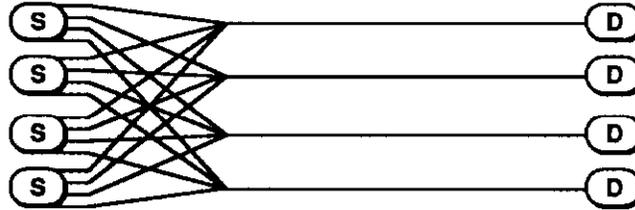


Figure 6: Multiport Memory Interconnection Network

This approach is still limited by the number of physical ports which can be supported by a module. To allow greater expansion, multiport memories can be further subdivided into an array of independent dual-port buffers as shown in Figure 7. Dual-port memory is easier to implement since it is available in the form of commercial integrated circuits (dual-port static RAMs, FIFOs or video DRAMs).

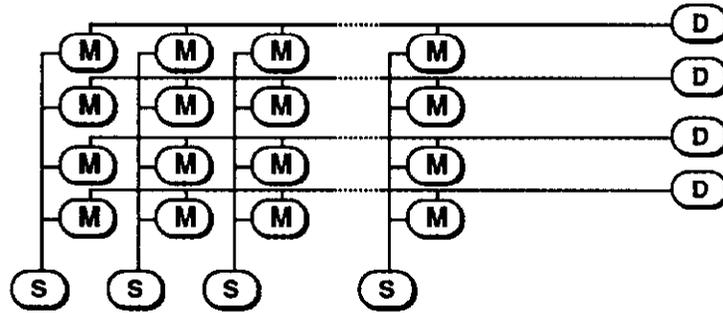


Figure 7: Dual-Port Memory Interconnection Network

With dual-port memory, the limitation now becomes the total number of buffers required in a larger system instead of the number of connections per buffer. The number of buffers can be reduced by using higher speed output busses, (allowing a rectangular instead of square array) or possibly by implementing some kind of multistage memory architecture (see references to Clos networks later).

In the dual-port memory architecture, the fragments of a given event are transmitted in parallel from the front-end subsystems to buffers in a selected row. These fragments are then read out sequentially by a processor while the next event is being transmitted to another row of buffers.

An example of a Multiport Memory Interconnection Network is the D0 data acquisition system at Fermilab as shown in Figure 8. D0 is really the reverse of the system described in Figure 6; each of N sources is connected to each of N destinations. Event building consists of bringing together the data from the Level 1 Trigger subsystem

and seven detector readout subsystems housed in VMEbus into one of several MicroVAX [8] computers. Data flows from VMEbus-based front-end electronics over eight cables which are in turn bussed to multiport memories in each MicroVAX computer of the online processor farm. In this experiment, the online processor farm is called the Level 2 subsystem. If an event is accepted after a Level 2 filtering algorithm, it is passed to a "host" VAX computer over another input/output (I/O) cable. A special feature of the D0 multiple bus system is the use of multiport memories, which eliminate the overhead associated with unnecessary copying of data within the system. Events arriving at one port of each of eight multiport memories in each MicroVAX computer are copied to internal memory using the computer's internal bus. This multi-port memory architecture allows events to be transmitted in parallel to the MicroVAX computers. However, the bottleneck is likely to be the three Megabytes/second transfer capability of the internal bus in the MicroVAX.

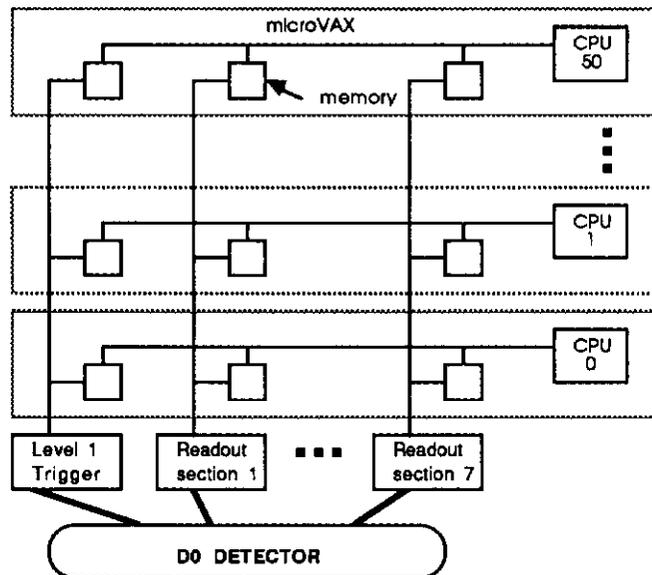


Figure 8: D0 Data Acquisition System

Crossbar and Other Switch-Based Interconnection Networks

The dual-port memory architecture in Figure 7 is actually a form of buffered crossbar switch. A crossbar switch provides a complete, non-blocking interconnection between all inputs and outputs. It is an ideal interconnection network in terms of bandwidth efficiency. Crossbars used in packet-switching networks can be classified by the location of the buffering (input, output or embedded) with respect to the switching matrix. If the buffers are moved to the inputs or outputs (Figure 9), the switching matrix itself can be confined to a very small area, usually inside a few VLSI circuits. As an added advantage, only $2N$ large dual-port buffers are required if the buffers are positioned at the inputs and outputs, whereas N^2 smaller buffers are required if they are embedded in the switching matrix. The total amount of memory required is the same regardless of where it is positioned, but as a practical matter it is easier and less expensive to implement a small number of large dual-port buffers compared to a large number of small dual-port buffers.

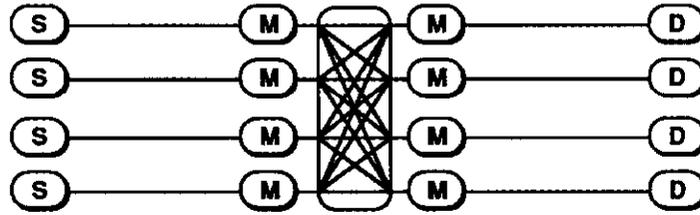


Figure 9: I/O Buffered Crossbar Interconnection Network

The full crossbar requires N^2 crosspoints, which may be impractical for larger systems, even in VLSI. A three stage Clos network, shown in Figure 10, is an example of a Multistage Interconnection Network (MIN). For systems with twenty or more data channels, a multistage network can provide essentially the same nonblocking characteristics as the crossbar switch, using fewer crosspoints.

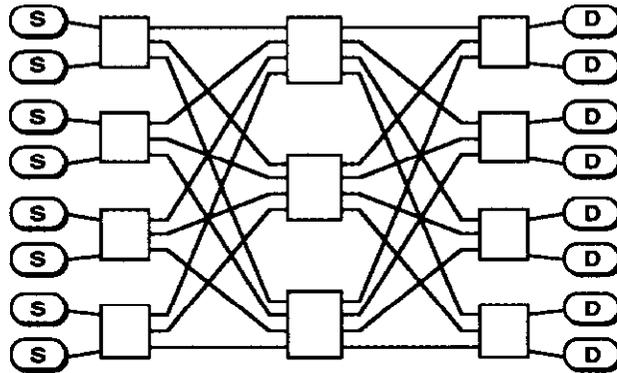


Figure 10: Three Stage Clos Interconnection Network

The number of the center stage switches is calculated to provide at least one more path than would be used if all inputs to a first stage switch and all outputs to a third stage switch (except the selected input and output) were busy. For an N channel system, the optimum switch sizes are $n \times k$ for the first stage, $N/n \times N/n$ for the second stage and $k \times n$ for the third stage, where n is approximately $\sqrt{N/2}$ and k is $2n-1$. In the example of Figure 10, $N=8$, $n=2$ and $k=3$. This yields a total of 96 crosspoints, which is actually more than the 64 required for a single stage crossbar. For a 512 channel system however, the three stage Clos network requires only one fourth as many crosspoints as the single stage crossbar.

In a large network, a centralized controller is often used to determine the best network configuration for a given interconnection pattern. For a random combination of sources and destinations, the time required to calculate this optimum switch configuration could far exceed the actual data transfer time. Fortunately, this level of control is not necessary in event building applications. Event readout follows a fixed access sequence which makes simple arbitration schemes very effective. At startup for example, all sources will contain a fraction of the data from the first event and will all arbitrate for the same output channel and processor. Only one is successful in transmitting its data, while the others are blocked. As the second event is read out, the source that was successful on the first arbitration now sends its portion of the second event to a different processor, while the remaining sources contend again for the first

output channel and processor. If events are nearly equal in size, the system will automatically converge to a state where each source is accessing a different output channel and there is very little contention. In fact, because this arbitration sequence is known beforehand, there is no real need to place arbitration circuitry in the network. The sources can be programmed to simply delay transmission by one time-slot with respect to the adjacent input, thereby avoiding contention altogether.

Because of this predictable access pattern, configuration control of a crossbar switch can be greatly simplified. In general, a complete random interconnect capability is not necessary as long as all inputs can be connected to all outputs at least once during the transmission of each event. A "barrel shifter" is a device which provides this simple rotating interconnection pattern using a single control input. The data multiplexing logic of an $N \times N$ barrel shifter is identical to that of a unidirectional $N \times N$ crossbar switch but with far fewer possible configurations (N versus NN). Barrel shifters are often used as the "space division" stage (physical circuit switching stage) of a time-division multiplexed (TDM) switching system. A crossbar switch can be operated as a barrel shifter by simply restricting the set of possible control inputs. This limited subset of available configurations is all that is necessary for event building, as illustrated in Figure 11. Here the events are assumed to be equal in length and evenly distributed. Non-uniform distributions of event data can be handled by the addition of Time-Slot Interchange (TSI) buffers on either side of the barrel shifter. TSI operation is explained in more detail later. This architecture closely resembles a typical telephone switching system (e.g., AT&T 4ESS or 5ESS [11]).

Figure 11 illustrates an idealized example of a four-input, four-output barrel shift switch where the size of all the input event data fragments (e.g., event number 1, fragments 1A, 1B, 1C and 1D) are not only equal but are equal to the packet length. Data passes through the switch in fixed-length packets with each input channel delayed by one packet time slot relative to the adjacent channel. With the switch control set to logic state 00, the first data packet (1A) passes directly through the switch along with three empty packets. The switch control is then incremented by one to logic state 01 and packets 1B and 2A are transmitted through the switch. During the next time slot (switch control set to logic state 10), packets 1C, 2B and 3A are transmitted. Finally, with the switch control set to logic state 11, packets 1D, 2C, 3B and 4A are transmitted. After one rotation of the switch control, the system reaches a steady-state condition. Parallel event fragments are converted to assembled event streams with no loss of bandwidth. Four packets of data from four different events cross the switch during each packet interval. The bandwidth of data flowing through the event builder matches the bandwidth of data from the detector.

The example given in Figure 11 is not only an idealized situation but shows the switch-based (barrel shift) parallel event builder IN working in an open-loop control mode whereby events are transmitted to sequential outputs. Final event destinations (processors in an online array of processors) are assumed to be ready to accept the next event. This mode of control for the barrel shift IN, along with three other modes of control, will be described in more detail later.

The simplicity of operation of this particular IN is possible only because of the nature of physics event building. Messages are unidirectional with predefined destinations. The control of the barrel shift switch requires only a counter, whereas dynamic calculation and loading of switch routing information during switch operation is required with a generalized crossbar. A final point in favor of the barrel shift interconnect involves the expansion capability. For example, a 1024×1024 barrel shift IN using currently available 64×64 integrated circuits would require only 32 ICs compared to 256 ICs for the equivalent crossbar.

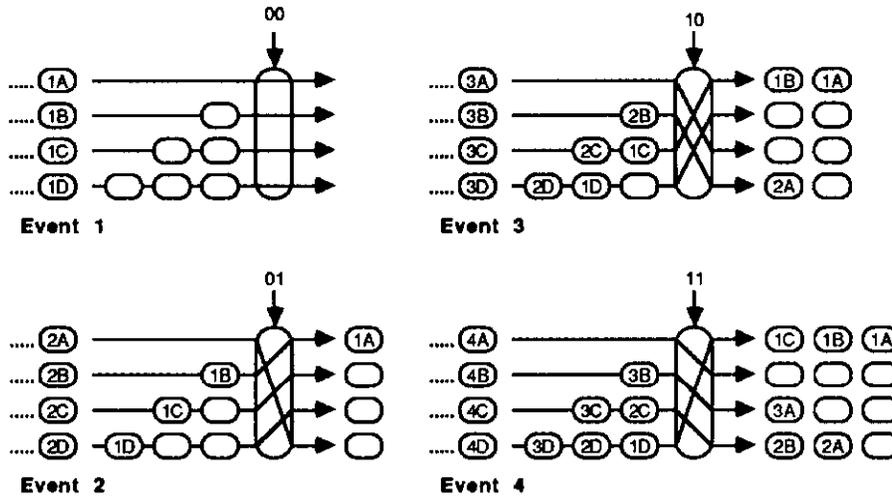


Figure 11: Barrel Shift Interconnection Network

Banyan or Delta networks are the basis of many self-routing packet networks. Under certain conditions, the total bandwidth of a Banyan network (Figure 12) matches that of a crossbar or Clos network, without the need for a centralized control mechanism. Each node of the network is a simple 2 X 2 switch designed for self-routing of data packets based on a destination header. To avoid blocking, this network is preceded by a sorting network or stage of time-slot interchangers which order the data packets in such a way that no contention for internal or external datapaths will occur.

Because there is active circuitry in each node of the switch, data rates are generally lower. Without internode buffers, the entire network must be bit and packet synchronous. This type of network will probably find use in future telephone switching applications as represented by the AT&T Starlite project [12], and could provide a commercial alternative to specially designed event builders.

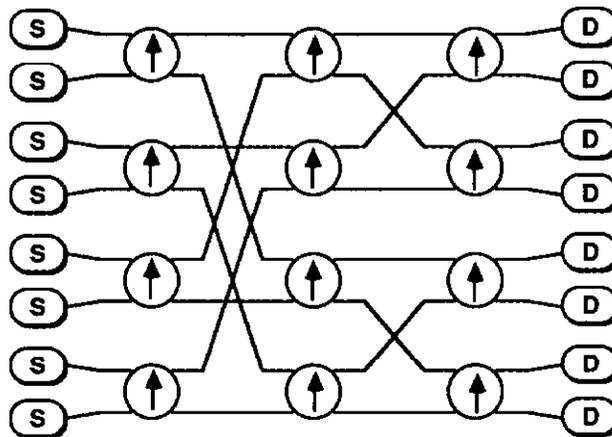


Figure 12: Banyan Interconnection Network

Integrated Processor Interconnection Networks

The mesh interconnection network (Figure 13) is popular in the construction of large multiprocessor systems (Intel Touchstone [13], Connection machine [14], Meiko transputer array [15]). These INs are formed by overlaying an array of processors on the dual-port memory array of the buffered crossbar switch. Some cost reduction may be possible with this approach. The mesh also allows direct processor to processor communication, not normally a requirement in event building but potentially useful in analysis of overlapping events or methods of event building which divide the analysis software into stages with each stage resident in different processors. Reliability can be higher for a mesh interconnect since there are multiple paths for each packet transfer. In practice though, the control complexity and possibility of message deadlock allows only orthogonal routing. Otherwise a packet may inadvertently be routed into a circular path and lost or delayed. Intelligent buffered routers are necessary for event builder applications because there is nearly continuous traffic on all links in the network. If the processors managed the internode communication directly, there would be little time left for processing the data.

The BCD detector collaboration at the SSC is investigating combining both the event building and online processor farm event reconstruction functions using a mesh interconnection network. It is hoped that by breaking down event reconstruction into a set of small functions, each of these functions can reside in the processors' cache memory thereby increasing the power of each processor. A high-speed mesh interconnection network would be used to transfer results from one processor to the next processor until the event has been both built and reconstructed.

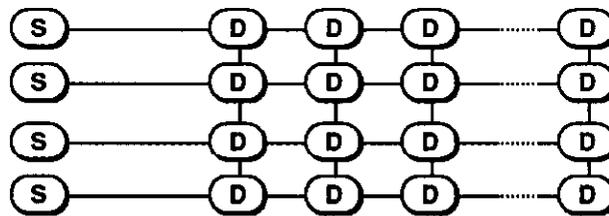


Figure 13: Mesh Interconnection Network

Communication Networks Applied to Event Building

A star coupler is a device used to connect computers and computer peripherals to one another. In the past the communication path has been over relatively slow data links such as Ethernet. With the advent of fiber optic technology and emerging fiber data link standards such as the 125 Megabit/second Fiber Distributed Data Interface [16] standard (FDDI), the one Gigabyte per second High Performance Parallel Interconnect standard (HPPI; formerly "High Speed Channel") [17], and the one Gigabyte per second fiber channel of the Scalable Coherent Interface [18] standard (SCI), star couplers are being developed with throughputs which will allow them to be used for many high energy physics experiments.

An example of a commercially available star coupler is the Ultra Network Technologies, Inc. [19] UltraNet 1000 Hub. Because fiber data link standards have not yet been finalized and integrated circuit support for these data link standards is not yet available, this company developed its own proprietary fiber data link and a modular 44 I/O channel device to get an early lead in the high performance, high throughput star

coupler market. This product, with its fifty Megabytes/second per I/O port effective throughput, is the highest performance commercially available star coupler network today. The performance is attained by putting software intensive communication's protocols into silicon. The product is a packet-based multiple but not parallel bus interconnection network with little buffering (i.e., non store and forward mode operation). As shown in Figure 14, the UltraNet 1000 Hub has a single, one Gigabit/second primary bus and multiple, one Gigabit per second local busses (one for each quad I/O port module). At the local bus level, each I/O port module supports two simultaneous input to output links each capable of fifty Megabytes/second throughput. Thus, the maximum effective throughput of this star coupler is 1.1 Gigabytes/second (i.e., $\{50 \text{ Megabytes/second} \times 2\} \times 11 \text{ quad I/O port modules}$).

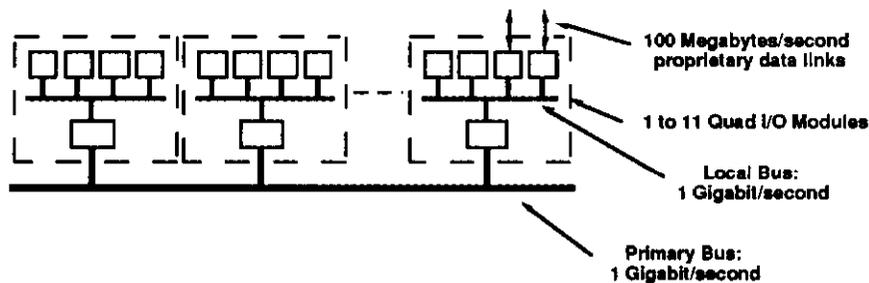


Figure 14: UltraNet 1000 Network Hub

Figure 15 illustrates the use of the UltraNet 1000 Hub as an event builder in a high energy physics experiment. Since each detector data source must pass through the Star Coupler to an online farm processor, the 1.1 Gigabyte/second effective throughput is not realizable. As shown in the Figure, given an experiment with forty data sources and two output links to online processors, in this case Silicon Graphics, Inc. workstations [5], the limiting factor in effective throughput is that event data from each data source must pass through the primary one Gigabit/second backplane. Including various overheads (e.g., communications protocols, arbitration times, etc.), the effective throughput through this star coupler when all messages must pass over the primary bus is approximately 50 to 80 Megabytes/second (~50% of 1/8 Gigabytes/second).

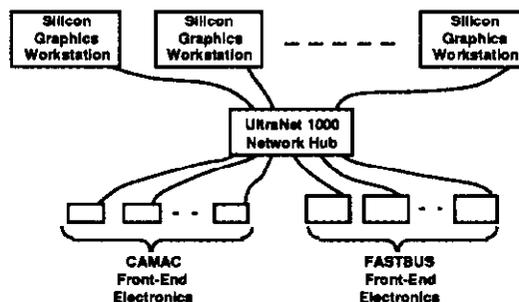


Figure 15: Data Acquisition System Using a Star Coupler as an Event Builder

The main point in mentioning this technology for possible event builder applications in high energy physics applications is not necessarily the tens of Megabytes/second effective throughput realizable today with a commercial product but the explosive growth potential of this marketplace and the use of standard industry supported data links for the transmission of data. Using a switch-based interconnection network or

multiple primary backplanes instead of a single primary backplane and using Gigabit/second or even faster standard data links, within five to ten years a large, commercially-available Star Coupler might have an effective throughput approaching the few Gigabytes per second needed for some SSC data acquisition systems.

ARCHITECTURAL CONSIDERATIONS

Several architectural considerations are common to all systems:

Mixed Control/Data Paths: This has become an obvious weak point in many existing systems. Very few high bandwidth data acquisition architectures would now consider mixing control and data on the same physical network.

Push/Pull Data Transmission: A "pull" architecture implies a bidirectional datapath and some limited mixing of control and data. A "push" architecture implies greater intelligence at the source and increased buffering at the destination. However, it also allows the use of high bandwidth, unidirectional data channels (e.g., fiber-optics) and a loosely coupled control structure. At high data rates and greater source/destination distances, "pull" architectures are not practical.

Centralized/Decentralized Control: There is always at least some centralized control in any data acquisition system. In particular, distribution of the low level triggers must be centralized to avoid moving unwanted data off the detector. Beyond the front-end, the need for centralized control is minimal. A global trigger rate control for the entire system or for individual output channels is sufficient, and does not seriously affect system throughput. All common control points should be located in one logical device (an obvious choice is the Level 2 trigger system). There is no need for separate centralized controllers at the detector, event builder and processing farm.

The ideal event builder architecture is one that provides the high bandwidth capabilities of a large IN, but without the complicated control mechanisms. Much of the complication in general-purpose INs result from the need to support random message traffic. A sophisticated controller is required, either centralized or distributed through the network, to avoid contention. In event building, much of this control can be eliminated by partitioning the system so that the average data rate between any combination of source and destination is nearly constant. This is similar to the advantage gained in designing a parallel processor interconnect when the processors are running a fixed algorithm with known interprocessor communication requirements.

A shared bus provides a very simple control mechanism, but must be eliminated because of low bandwidth. Expanding to a multiple bus or multiple port memory architecture is physically awkward for more than three to eight channels.

The dual-port memory array (crossbar with embedded buffers) is a good choice for systems with up to 32 channels, after which it becomes large and somewhat expensive. Moving the buffers to the inputs and outputs of the crossbar allows the system to expand linearly, but requires some additional control. Restricting the interconnection pattern (e.g., barrel shift instead of full crossbar) reduces the size of the switching network. Multistage networks are more efficient, in terms of number of crosspoints, but also more difficult to configure. The availability of VLSI crossbars and the limitations on data acquisition system size (typically less than 256 channels) make multistage networks unnecessary.

Some of the self-routing packet networks now being investigated for telecommunications use are overly complicated for event builder needs. This additional complication may be offset by the advantages of buying a standalone commercial product.

ADDING FAULT TOLERANCE TO INTERCONNECTION NETWORKS

A fault tolerant interconnection network is one that provides service, in at least some cases, even when it contains a faulty component or components. A network is "single-fault tolerant" if it can function as specified by its fault-tolerance criterion despite any single fault conforming to its fault model[20]. A network is "i-fault tolerant" if any set of "i" faults can be tolerated. A network that can tolerate some instances of "i" faults is "robust" although not "i-fault tolerant".

Two methods used to add fault tolerance (redundancy) to interconnection networks are dilation and replication [21]. Dilation, as shown in Figure 16a, expands or "dilates" an IN stage or stage subsection. If one path through an IN stage subsection fails, an alternate path through the same stage subsection is used. Replication, as shown in Figure 16b, does not alter the IN stage subsection but adds identical or "replica" stage subsections. Additional stage subsections are used when their duplicate stage subsections fail. Both dilation and replication improve system performance (by reducing the probability of blocking) and reliability at a cost of increased price and complexity. A message arriving at any input of a 4 x 4 switch in Figure 16a or a 2 x 2 switch in Figure 16b can be switched to any output. In both figures, normal message paths are shown in bold.

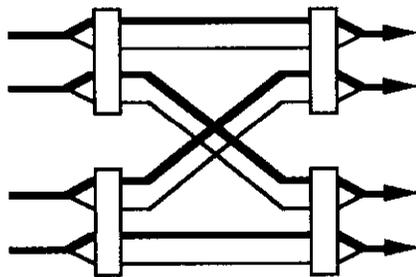


Figure 16a:
Dilation Redundancy

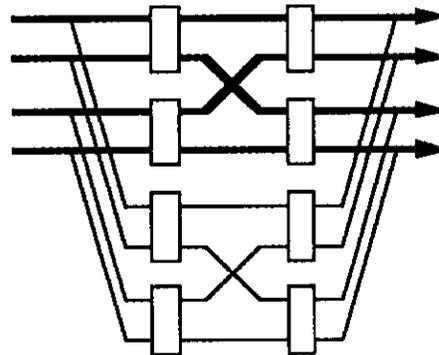


Figure 16b:
Replication Redundancy

Another less obvious method for adding fault tolerance is by using an "extra stage cube" network [24] as shown in Figure 17. For each 2 x 2 switch, messages arriving at port "a" or port "b" can be sent to either port "c" or port "d" or to both ports "c" and "d" simultaneously (i.e., broadcast). The first (leftmost) and fourth (rightmost) stages can be enabled or bypassed. The first stage is enabled when it is being used and not "bypassed" by the multiplexers shown after each first-stage 2 x 2 switch. The fourth stage is enabled when it is being used and not "bypassed" by the demultiplexers shown before each fourth-stage 2 x 2 switch. Normal operation of the IN is with the first stage enabled and the fourth stage bypassed. If a fault occurs in the fourth stage, no reconfiguration of the IN is necessary. If a fault occurs in the first stage, it is disabled and the fourth stage is enabled.

If a fault occurs either in a link or in either of the inner stages, both the first and fourth stages are enabled. Multiple-fault tolerance is enhanced by individually enabling and disabling the first and fourth stage multiplexers and demultiplexers, respectively. This network is said to be "single-fault tolerant" and "robust" in the presence of multiple faults.

In a specific application in high-energy physics, a decision whether to use either of these methods of fault tolerance would have to be based on the ease with which failures are diagnosed and repaired in a system without fault tolerance versus the added price and complexity of implementing and maintaining a system with fault tolerance.

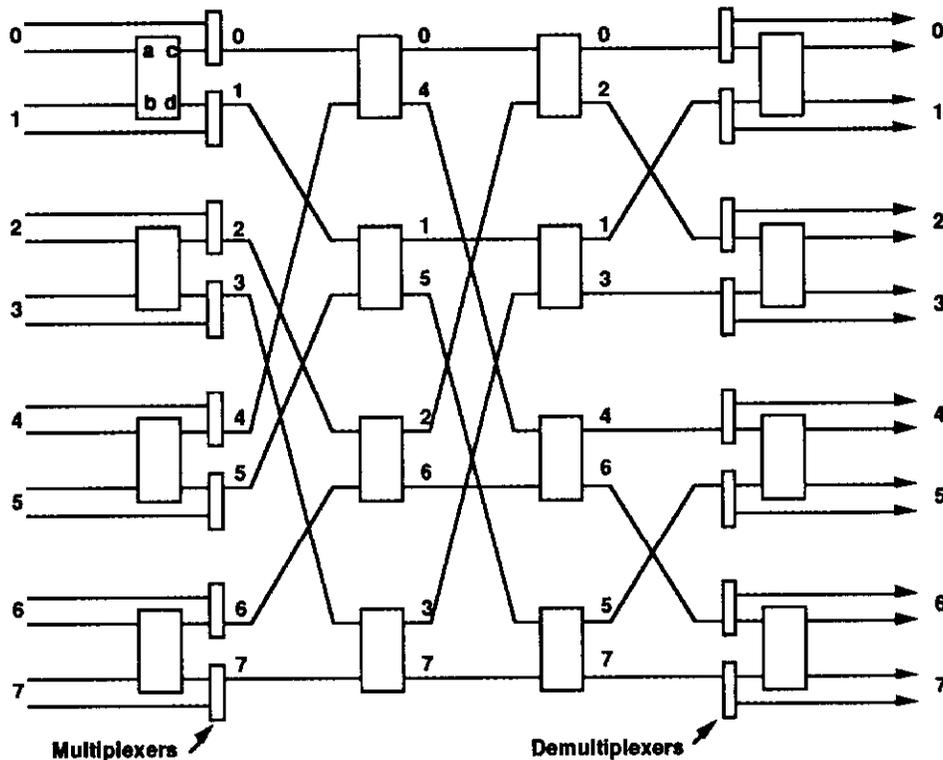


Figure 17: Extra Stage Cube 8 x 8 Fault Tolerant Interconnection Network

SELF-ROUTING TECHNIQUES

Routing tags placed in message headers are used to describe a path through a self-routing network. In a fault-tolerant self-routing network, these tags specify a functioning path. There are three methods for sources to generate routing tags that specify a fault-free path. With "non-adaptive" routing, a source is notified of a malfunctioning path when a message it has initiated reaches a faulty component. This approach requires little hardware but usually has poor performance. There are two forms of "adaptive routing". With "notification on demand" adaptive routing, a source maintains a table of faults it has encountered while attempting to establish paths. This table is used to derive routing tags for subsequent messages. With "broadcast notification" adaptive routing, each source is notified of any fault encountered by any message attempting to establish a path. With another method of routing, "dynamic routing", routing tags are "dynamically" altered as messages pass through a network and faults

are encountered. This routing method will not be discussed any further in this paper. A few techniques for self-routing are discussed below.

With circuit-switched INs, a complete path from source to destination is established prior to the initiation of message transmission. No buffers are required in any internal IN stages. With packet-switched INs, packets of fixed or variable sizes with routing tag headers are sent to and buffered in each succeeding stage a message passes through in the IN. Only links between two switches of two adjacent stages need to be established at any one time. Messages are "stored and forwarded" from (switch) stage to (switch) stage until reaching their destination.

Wormhole routing differs from packet-switched routing in that only one word of a packet is forwarded to the next switch after the current switch has received and latched the next word of a message. Less buffering is required than in a packet-switched IN. Message transmission is halted if a downstream switch is busy passing another message. Messages are thus "pipelined" through the IN.

Virtual cut-through routing is similar to wormhole routing except that when a message gets blocked at a busy switch, the remainder of the message is transmitted to and buffered in the busy switch. More buffering than in wormhole routing is required but effective throughput is increased by not keeping all upstream switches in a blocked message busy until the message is no longer blocked.

Figure 18, an 8 x 8 Multistage Cube IN, will be used to describe three methods for defining routing tags in this self-routing IN. In all the examples, the message source ID is binary six (110) and the destination ID is binary three (011). Each 2 x 2 switch has four operating modes as shown at the bottom of the figure. Broadcast modes will not be discussed.

With the first method, the routing tag is the destination (011). At each stage, the switch receiving the message examines its component of the routing tag (i.e., stage 2 examines the 2^2 bit, stage 1 the 2^1 bit, etc.) to determine how to route the message. In a 2 x 2 switch, the upper input port is port 0 and the lower input port is port 1. Routing is determined as follows. If the switch's component of the routing tag is logic "A" and the input port receiving the message is port "A", the switch operates in the "straight" mode as shown in the figure; if the port receiving the message is port "not A", the switch operates in the "exchange" mode.

The second method uses the rule that, with a 2 x 2 switch, the upper port is port 0 and the lower port is port 1. The switch simply uses its component of the routing tag, the destination as in the first method, as a pointer to output port 0 or output port 1. Both this and the previous method allow verification by the destination that it was supposed to receive the message (i.e., destination ID equals routing tag).

With the third method of routing, the routing tag is the logical bitwise "exclusive or" of the source and destination. If an input port of a 2 x 2 switch receives a message and its component of the routing tag is logic 0, the switch operates in the "straight" mode; if logic 1, the switch operates in the "exchange" mode. The disadvantage of this self-routing technique is that the routing tag is more difficult to compute. The advantage is that a destination can derive the source of a message by doing an "exclusive or" of its address, the destination, and the routing tag.

Combinations of the above self-routing techniques, along with error detecting and possibly correcting codes on the routing tag and even the data, allow the destination not only to identify the source of the message and to determine if it was supposed to receive the message but also guarantee data and message integrity.

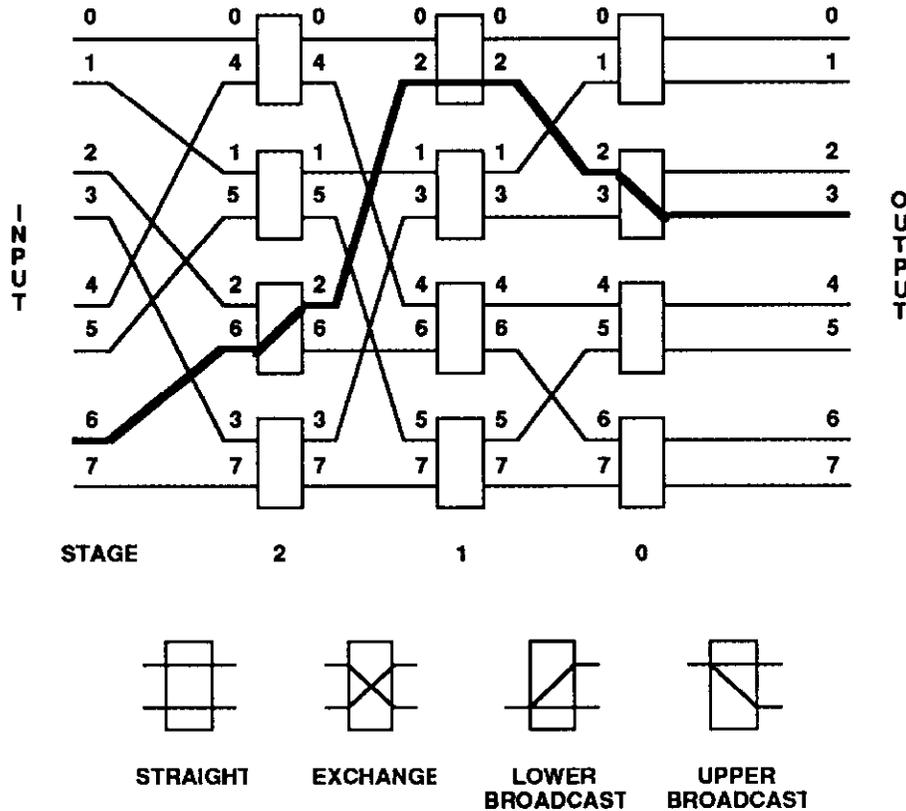


Figure 18: Multistage Cube Interconnection Network

INPUT AND OUTPUT QUEUEING IN AN INTERCONNECTION NETWORK

Packet switching networks often employ input or output queues to regulate data flow in the network [22]. Most methods of queueing assume that packets are fixed-length and that each packet has an equal probability (i.e., $1/\text{number of outputs}$) of being addressed to any given output. Input and output queueing implies FIFOs at the IN inputs and outputs, respectively.

With input queueing, the number of message packets with the same destination sent through the IN during any one time slot is controlled. Potential bottlenecks at the IN outputs are minimized and throughput is actually increased even though the transmission of some message packets is delayed. The disadvantage is that some packets which could have been transmitted to an idle output are blocked by a preceding packet which is waiting for a different, busy output.

With output queueing, it is assumed that the internal network links can operate at a much higher bandwidth than the input or output channels. Packets arriving simultaneously at the same output are queued until the output is ready. Output queueing is more efficient than input queueing because there is no blocking within the network itself. However, the assumption that the network links can operate at N times the single channel I/O bandwidth is not very realistic.

Time Slot Interchangers provide the advantages of output queuing while being physically located at the input of the IN. They act to resequence the input data so that no packet is blocked by preceding packets. This is equivalent to the input queuing model with a separate FIFO for each destination. Various other combinations of input, output and internal queuing (not mentioned in this paper) are also possible.

**A FUTURE DATA ACQUISITION SYSTEM
ARCHITECTURE & SOME FUTURE TECHNOLOGIES**

Specified event building data rates for some of today's existing, under development and proposed high energy physics experiments are given in Table 1. Note the two to three orders of magnitude increase in data rates in two of the proposed Superconducting Super Collider (SSC) detectors from present-day experiments. This is due to expected particle interaction rates of 10^8 and 10^7 per second, respectively, for the Solenoid and BCD detectors. It is obvious that new techniques of event building, most likely entirely parallel, need to be developed.

<u>Experiment</u>	<u>Event Building Data Rate</u>
ALEPH (CERN)	1 Megabyte/second
DELPHI (CERN)	2 Megabytes/second
L3 (CERN)	8 Megabytes/second
CDF (Fermilab)	15 Megabytes/second
D0 (Fermilab)	27 Megabytes/second
Solenoid (SSC)	1-10 Gigabytes/second
BCD (SSC)	10-100 Gigabytes/second

Table 1: Present and Future Event Building Data Rates

Figure 19 illustrates data acquisition and triggering data flow requirements for the proposed large solenoid detector at the SSC. 60 MHz beam crossings and 100 MHz interaction rates are reduced to trigger rates of 1 KHz after two levels of triggering. Substantial intermediate event data buffering during triggering is required. With average event sizes of one Megabyte, event data must pass from the intermediate buffers through a parallel event builder into an online processor farm (L3 Farm) at an average rate of one Megabyte every one millisecond, 1000 events/second or one Gigabyte /second. Designing the data acquisition system with a factor of ten higher throughput capability for future growth and possible higher trigger rates and larger event sizes, requires a ten Gigabyte/second average throughput parallel event builder.

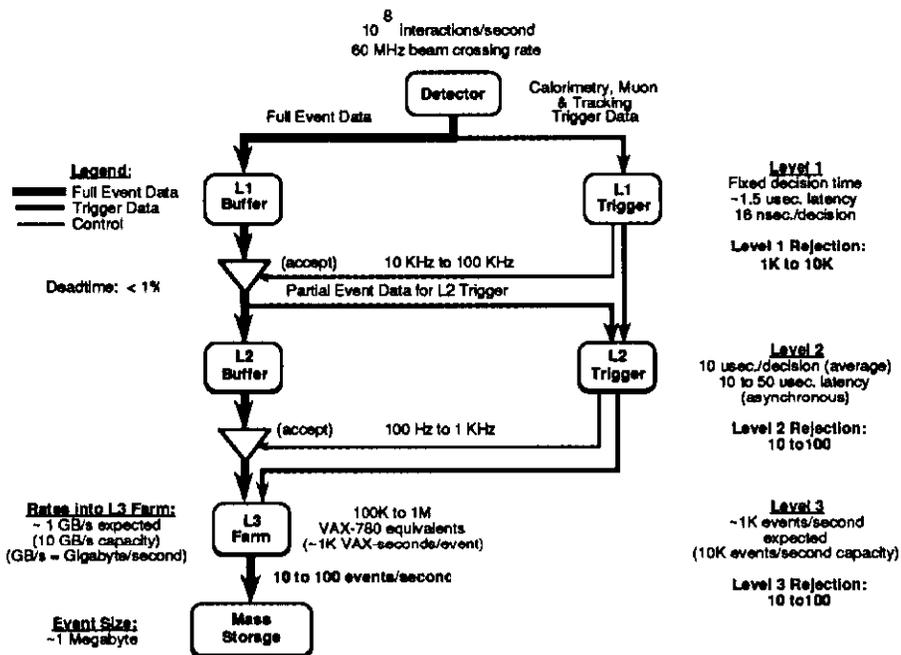


Figure 19: Trigger & Data Acquisition Dataflow for the SSC Solenoid Detector

Figure 20 illustrates a proposed new data acquisition system architecture for the two SSC detectors. Two levels of triggers are proposed for the Solenoid detector; only one level of triggers is proposed for the BCD detector. The central component of the architecture is a parallel event builder or interconnection network. Extensive system simulations are needed to define the parallel event builder to be used on these detector data acquisition systems. The brief introduction to interconnection networks and various technologies described in this paper should aid in choosing an implementation method for the parallel event builder.

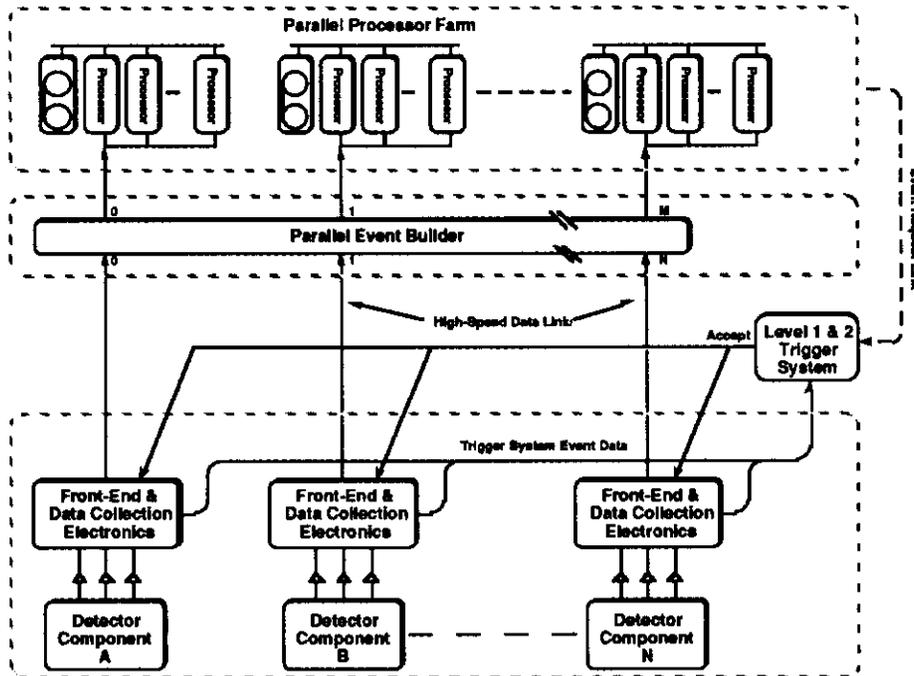


Figure 20: Proposed Data Acquisition System Architecture

Front-End Architecture

To achieve the needed high data rates through a parallel event builder in an SSC experiment, much of the front-end electronics needs not only to be mounted on the detector but needs to contain buffers for several events. Data and triggers must be pipelined to eliminate deadtime. Front-end electronics will contain mostly analog pipelined buffers to store data for a few microseconds at the sixteen nanosecond crossing rate of the detector during Level 1 triggers. Data will be stored for several tens of microseconds in analog or digital buffers during Level 2 triggers. A possible front-end and near detector architecture is shown in Figure 21. Standard readout ICs, Data Collection ICs, would also be mounted on the detector and would be used to read data from all front-end subsystems. For most and possibly all subsystems, event data will be stored in front-end ICs until after Level 2 triggers. Because of the extremely high event rates and widely varying distributions of data for a particular event within various front-end ICs, all of one event's data will not necessarily be received at the Data Ordering logic before the data from events occurring later in time. Thus, the Data Ordering logic is a temporary buffer for several events. A parallel event builder operates most efficiently when approximately equal amounts of data arrive at each of its inputs averaged over several events. Data Balancing logic is used to help equalize the distribution of data being transmitted to each input of the parallel event builder. High-speed fiber data links are used to transmit data to the parallel event builder.

Another approach to event building involves sending individual fragments of event data through a multistage interconnection network as they are read by the Data Collection ICs, without waiting until all local data for a specific event is collected. Both this technique and others described above need extensive simulation before the proper design can be decided upon.

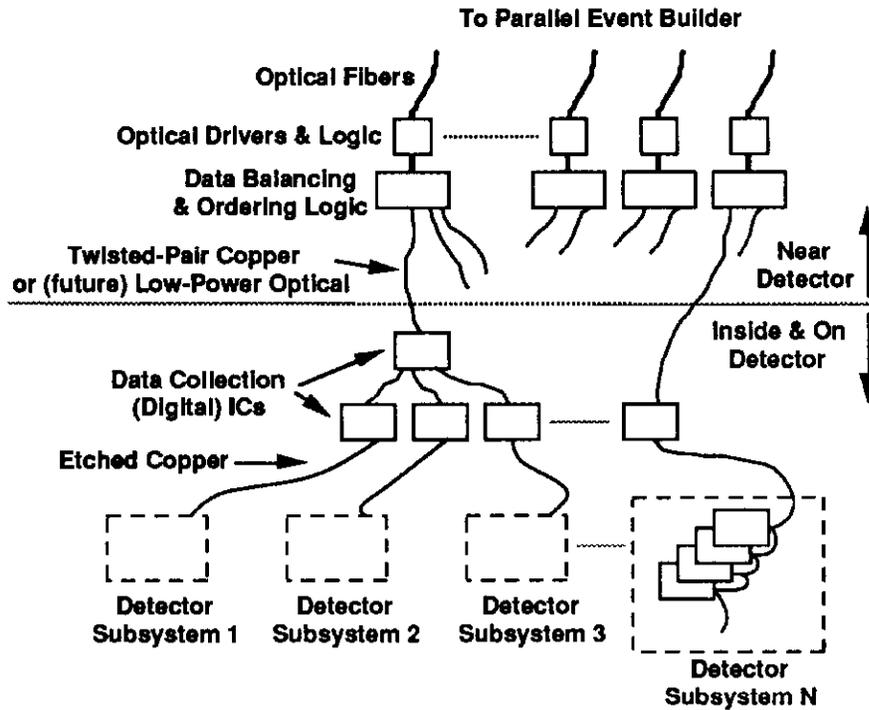


Figure 21: Possible Future Front-End & Near Detector Architecture

Pre-Processing

A major new technology for interconnecting multiple IC wafers with very large numbers of interconnections is "3-D" packaging (being developed by Hughes Aircraft [23] and others). As shown in Figure 22, the technology stacks several silicon wafers separated by spacers, interconnects the wafers by thermally dissolving molten droplets of aluminum such that they "eat" through a wafer, and connects the wafers to the outside via conventional flat cable. A 32-input, 32-output five-wafer stack, having well over 4000 feedthrough interconnections between the wafers, and able to withstand shock tests needed for military use has been successfully tested by Hughes. Each feedthrough has approximately twenty ohms of resistance. Present development plans include a "3-D computer" consisting of a 128-input, 128-output 15-wafer stack with nearly 250,000 interconnections operational in 1990 and another consisting of a 512-input, 512-output 25-wafer stack with over 1,000,000 interconnections operational in 1994.

Data presently is fed into the wafer stacks in a digital serial format at relatively few tens of Megabits/second. The physical properties of the flat cable will limit the data rates on the I/O cables. In time, different cabling technologies will be used, both increasing I/O data rates and allowing analog and digital I/O. Wafer stacks under development presently contain only digital circuitry but there is no reason totally analog or combined analog/digital wafers couldn't be used.

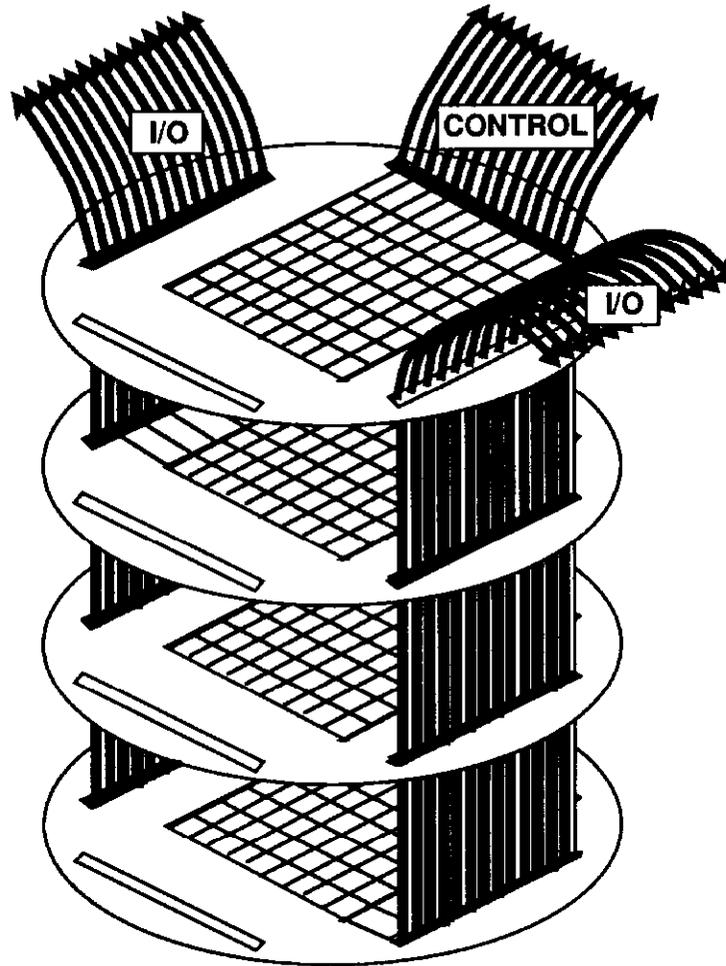


Figure 22: Three-Dimensional Integrated Circuit Packaging

In high-energy physics, these wafer stacks have several applications such as track segment finding, silicon pixel preprocessing, combined preprocessing (i.e, calibrations) and event building, etc. Their compactness make them ideal for installation right on detectors. Their interconnectability makes them very suitable for interconnection networks and preprocessing at various stages of a data acquisition system. For example, top wafers could be partially processing physics event data while lower layers are receiving data on the flat cables to further process this data with the results of the top wafer stages of processing.

Data Links

The transfer of high-speed serial data over wire and/or fiberoptic cable will be required in many future data acquisition system architectures. Several commercially available VLSI chips such as the Advanced Micro Devices TAXI [24] and Gazelle HOT ROD [25] integrated circuits appear to be useful for these applications. Future Local Area Network (LAN) and data link standards such as FDDI, HPPI and SCI must be studied to determine if they are appropriate to the proposed DAQ architectures. The major cost item in a fiberoptic data link operating at 250 Megabits/second or higher is the optical driver and receiver. The development of low-cost, high-speed (500 to 1000

Megabits/second) optical components represents a major effort. Thus far, the telecommunications industry has only concentrated on high-power optical components capable of signal transmission without repeaters over distances of miles.

Parallel Event Builders

Research on optical or opto-electronic switching systems can be directly applied to parallel event builder design. A very good overview of current work is presented in [26]. The use of AT&T's recently developed self-electrooptic effect device (SEED) in high-speed switching networks, along with general information on optical switching is covered in [27] and [28].

Figure 23 illustrates two approaches to optical switching, using optical shutters and waveguides to form crossbar switches. Switches of this type have been proposed or implemented with up to 32 channels. Although a fully optical datapath would seem to have advantages over a system which converts from optical to electronic and back, there is the potential problem of resynchronizing the optical receivers for each change in transmitters. A more likely candidate for larger switches is opto-electronic integrated circuitry (OEIC) where the inputs and outputs are optical fiber, but the actual switching logic is conventional GaAs. Some decoupling of the inputs and outputs could take place in the electronic part of the switch so that the individual links remain synchronized.

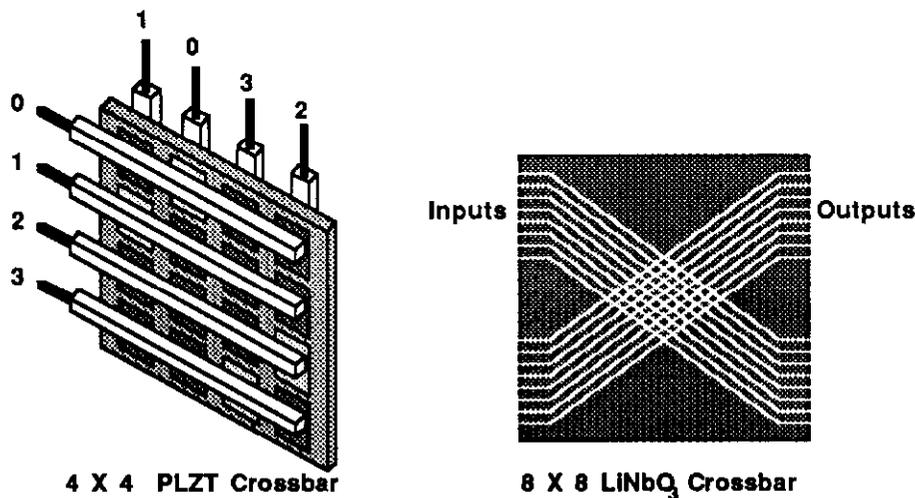


Figure 23: Optical Switching

Online Processor Arrays

To be cost-effective, future general-purpose processors must be highly integrated. Several major semiconductor manufacturers predict the availability of 50-100 million transistor ICs by the year 2000 [29]. Figure 24 shows an example of the type of general-purpose architectures which will be made possible by this level of integration. Since most on-line physics applications fit easily into an eight Megabyte memory space, a single IC multiprocessor is ideal for use in a high level processor farm. Even with conservative specifications (80 nanosecond DRAM, 50 MHz clock), an eight processor IC could deliver 250 MIPS at less than \$5/MIP. This is the target cost/performance range which would

allow the use of general-purpose processors in a "million VAX-equivalent" processor farm.

The Intel Touchstone project [13] is an example of near-term technology in processor farms. This system is expected to deliver approximately 200,000 VAX-equivalents (floating-point) in a 2048 node configuration by 1992 and should be scalable to the "million VAX-equivalent" range by the late 1990s. The Touchstone architecture uses a mesh interconnect, but allows direct routing between any two nodes without store-and-forward buffering. The network could be used for event building, as well as processing, if it were preceded by a stage of TSIs. Without TSIs, a significant portion of the processor local memory and I/O bandwidth may be needed for store-and-forward buffering since all input messages are contending for the same destination.

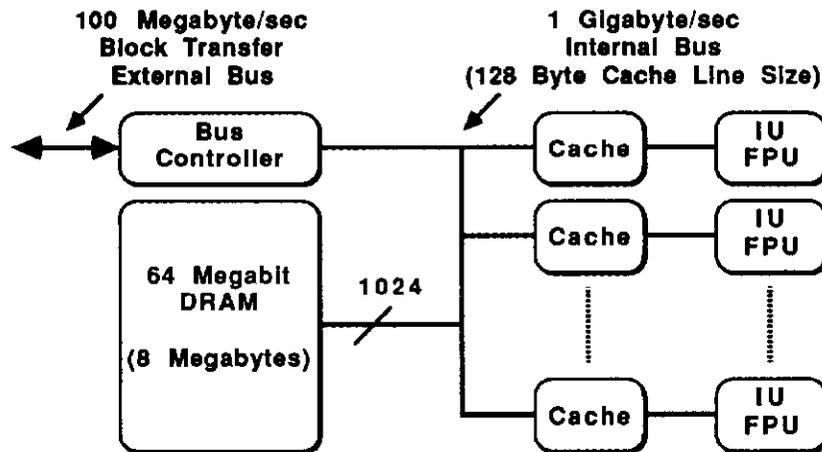


Figure 24: Integrated General-Purpose Multiprocessor

FERMILAB'S DATA ACQUISITION SYSTEM ARCHITECTURE & PARALLEL EVENT BUILDER PROTOTYPE PROJECT

A new data acquisition system architecture called the Scalable Parallel Open Architecture Data Acquisition System [30], is being developed at Fermilab. The goal of the project is to build a prototype system whose central component is a switch-based self-routing parallel event builder [31]. In order to test the system, a crate of test modules representing physics event data sources will be used to provide high speed parallel inputs to the switch, while at the outputs of the switch a crate of electronics representing an online processor farm will receive high speed "built" events in parallel. Extensive behavioral modeling and simulation experiments are presently being undertaken with the goal of understanding not only individual components of the architecture but also how they interact in the system. The architecture is scalable, firstly, in so much as it is well suited for data acquisition systems in low to high-rate experiments, test beams and all SSC detectors. Secondly, as both technology and physics needs change, the architecture "scales" for higher throughput (by adding more channels and/or processors) without modifying the fundamental structure of the system. This last feature is also implied in the "Open Architecture" part of the project's name. "Open architecture" means that new technologies (e.g., new online processors from several companies, newer and faster data links, etc.) can be added to the system (or replace existing elements) with little extra system development required. The prototype system at Fermilab will contain up to 64 channels, each operating at a nominal twenty Megabytes/second rate for a

combined throughput of approximately one Gigabyte per second. The parallel event builder is implemented using a barrel shift switch packaged in a single 9U Eurocard VMEbus crate and is expandable for higher data rate or additional data source requirements. The detector and processor farm are both emulated by test modules which transmit and receive simulated event data at full bandwidth.

Switch-Based Parallel Event Builder Operation

The main component in this new data acquisition system architecture is the barrel shift switch, parallel event builder interconnection network (IN). It can be classified as a packet-switched, synchronous, centralized three-stage IN. The input of the first stage and output of the third stage operate in an asynchronous mode in that data are stored at the input of the IN at random intervals using FIFOs and totally assembled events are transmitted from the outputs of the IN asynchronous to the functioning of the switch component of the IN. Event fragments are synchronously transmitted in packets through the switch or middle stage of the IN by centralized control electronics. This parallel event builder can operate in either self-routing or non-self-routing modes. In the self-routing mode, input event fragments are received, then "tagged" with their final (processor) destination address. In the non-self-routing mode, input event fragments pass through the switch stage of the IN in the order they were received with no processor "tag". Totally built events are transmitted to successive banks of processors and are lost if no processor in a bank is ready to accept an event. More will be said about these and other modes of control later.

The operation of the barrel shift event builder IN can be best explained by describing the logical operation of the network. Figure 25 is an illustration of a four-input, four-output (4 x 4) parallel event builder. Tagged event data fragments arrive at each of the four inputs and are placed into "logical" FIFO buffers in the first stage of the IN, the Input Time Slot Interchangers [32]. There is one Input TSI for each input data source. One "logical" FIFO buffer exists for each output of the barrel shift switch. Each event data fragment entering the IN is placed into the FIFO buffer corresponding to the output port from which the particular event will be forwarded to a processor. Thus, if the IN is a 4 x 2 network, each Input TSI will contain two "logical" FIFO buffers, one for each IN output. The third stage of the parallel event builder IN, the Output Time Slot Interchangers, consist again of "logical" FIFO buffers, one for each input of the IN. The middle stage of the IN, the barrel shift switch, is depicted in this figure as parallel connections of Input TSIs to Output TSIs. Each FIFO buffer of each Input TSI has a single connection to a "mirror image" FIFO buffer in each of the Output TSIs.

Figure 26 is a simplified illustration of the physical implementation of the parallel event builder IN being developed at Fermilab. The "logical" Input and Output TSI FIFO buffers are being implemented using dual-ported video dynamic RAMs logically divided into N circular buffers. At the Output TSI, a full event is assembled by concatenating one event fragment from each of the N buffers as the data is transmitted to a processor. The barrel shift switch is implemented using a programmable crossbar configured as one or more independent barrel shifters to allow system partitioning. A small Programmable Array Logic device (PAL) could have been used.

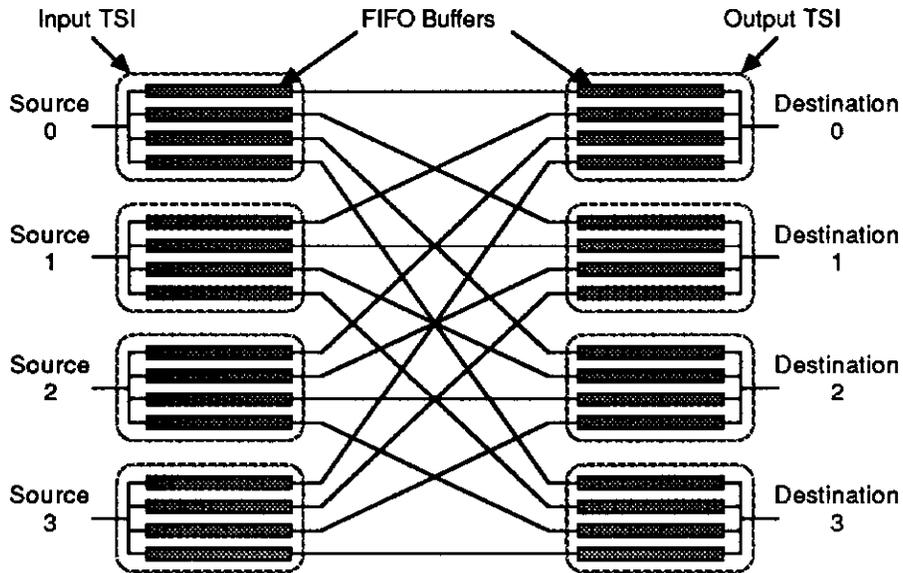


Figure 25: Barrel Shift IN Logical Operation

The barrel shifter rotates through its N possible states, connecting every logical input buffer to every logical output buffer once during a full rotation. In this way, the 32 logical buffers (four per TSI) and the sixteen logical interconnections shown in Figure 25 are emulated by a single physical buffer in each TSI and a four channel barrel shifter as shown in Figure 26. For a four channel system, the savings are not significant. But in a fully connected 1024 channel system, two million independent buffers and one million cables (plus connectors, etc) would be required. Using a switch-based architecture reduces this to 2048 buffers and 2048 cables.

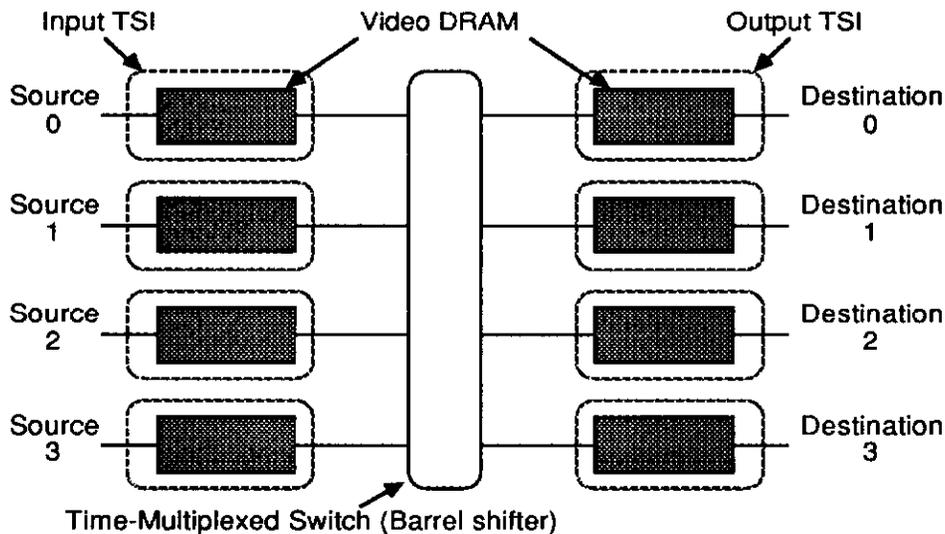


Figure 26: Barrel Shift IN Physical Implementation

Data crosses the switch in synchronous fixed-length packets. A single packet may consist of data from several events or a small part of a single event. There is no correlation between event and packet boundaries. The system emulates the logical operation of Figure 25 by moving small "time-slices" of data from the appropriate Input TSI buffers to Output TSI buffers based on the current interconnection provided by the barrel shifter.

N-Input & M-Output Barrel Shift Interconnection Network

As mentioned previously, the barrel shift switch interconnection network is not limited to N x N operation. A N-input, M-output (N x M) barrel shift switch IN differs from that of an N x N switch only in the number of "logical" buffers assigned (either dynamically or at system initialization time) to each Input and Output TSI. For the N x M switch, each Input TSI is assigned M "logical" buffers and each Output TSI is assigned N "logical" buffers.

Input & Output Time Slot Interchangers

The Input Time-Slot Interchanger packetizes the incoming data (event fragments) and rearranges these packets such that, for any configuration of the switch, each packet has a unique destination. This guarantees non-blocking operation of the switch and also serves to average the data rate on the input and output data links for better efficiency. The Output TSI concatenates event fragments to form a complete event for output to the processors. A single dual-port video DRAM is used in each TSI channel and is partitioned (through software pointers) into any number of logical buffers. The Input TSI can also direct incoming data to a specific output buffer based on a packet header. This provides the self-routing capability of the network.

The input and output TSIs are basically "mirror-images" of each other and reside on either side of the barrel shift switch which is implemented on a common backplane (Figure 27).

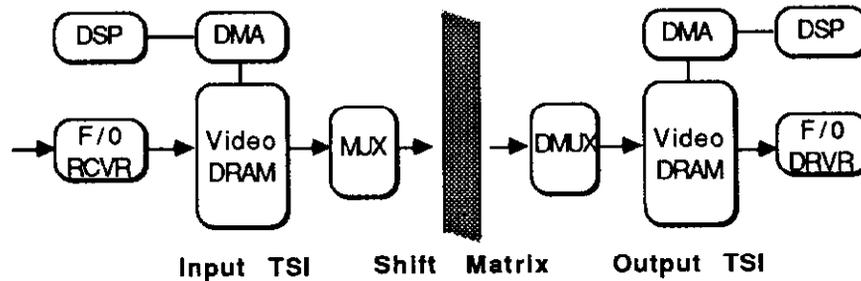


Figure 27: Time-Slot Interchangers

Switch-Based Parallel Event Builder Integration

Figures 28 and 29 illustrate how the barrel shift switch parallel event builder IN integrates into existing and new systems, respectively. In Figure 28, Input and Output TSIs are integrated with the switch stage of the IN. Asynchronous, non-packeted data are sent to the IN over fiber cables from detector event data sources and asynchronous, non-packeted data are transmitted to an existing array of processors. In Figure 29,

Input and Output TSIs are near the detector event data sources and the array of processors, respectively. Data packets are sent from remote Input TSIs to the switch stage of the IN and transmitted from the switch to remote Output TSIs over fiber and copper cables, respectively. Figure 30 illustrates what the future holds when using switch-based parallel event builders for high-energy physics experiments. Both the computer and telecommunications industries are developing opto-electronic integrated circuit (OEIC) switches or totally optical switches that should be usable in high-energy physics parallel event building applications.

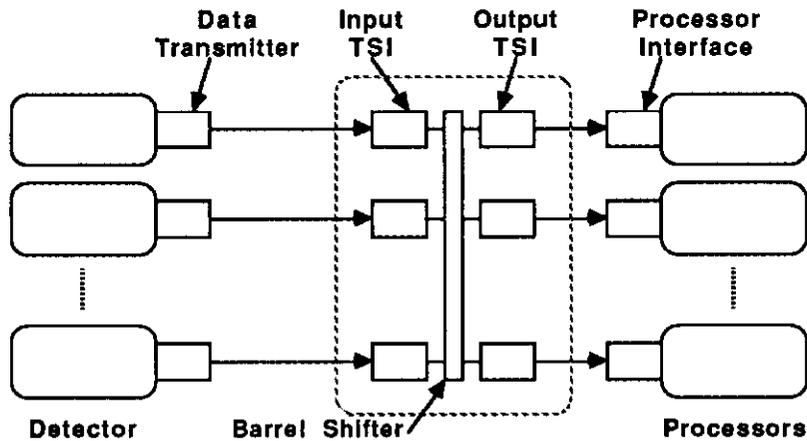


Figure 28: Integration Into Existing Systems

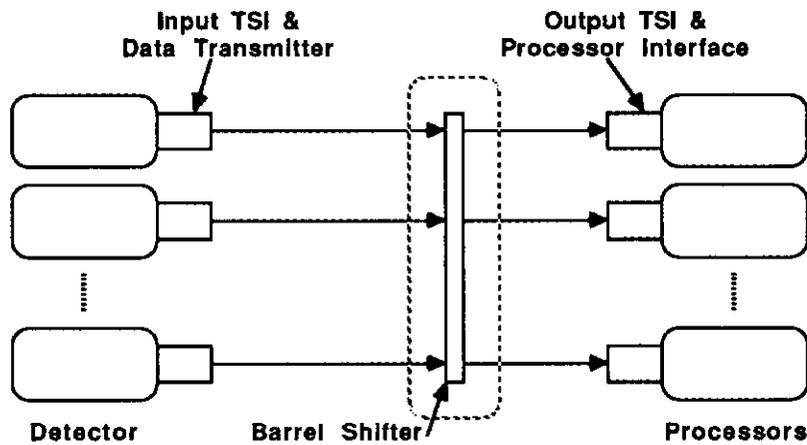


Figure 29: Integration Into New Systems

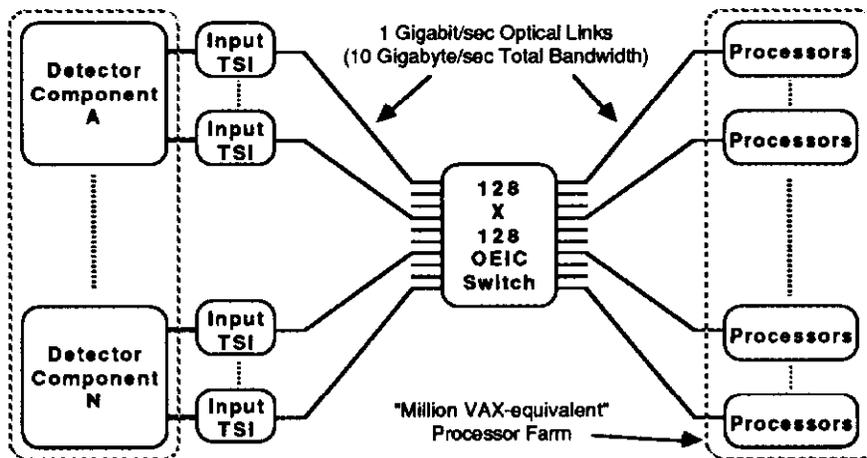


Figure 30: Possible Future Parallel Event Builder Configuration

Control Modes Of Operation

Four possible modes of operation of the barrel shift switch parallel event builder IN are being investigated, two open and two closed loop modes. Refer to Figure 31 to aid in understanding the four modes of control.

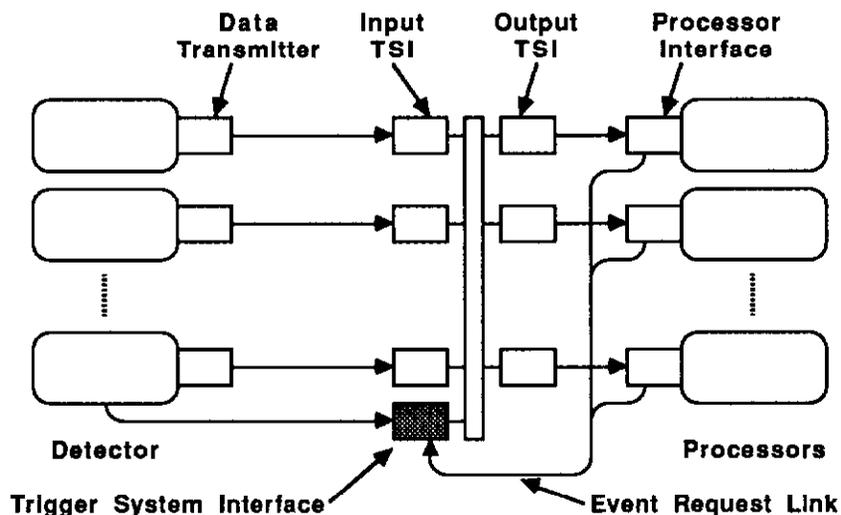


Figure 31: Event Request Link & Trigger System Interface

The first control mode, Open Loop Sequential, does not use the Event Request Link and automatically assigns events to the next sequential destination (i.e., output of the event builder). If a processor is not ready to accept an event, the event is lost. Accepted events can be delivered to the first available processor or to a specific processor designated by the event header.

The second control mode, Open Loop Non-Sequential, again does not use the Event Request Link and automatically assigns events to destinations based on stored

distributions of processors or strings of processors and processing power loaded into the Trigger System Interface at system initialization time.

The third control mode, Closed Loop Sequential, uses the Event Request Link to indicate whether a destination processor is ready. The Trigger System Interface assigns events to the next sequential ready destination. Event triggers are disabled if there is no ready processor connected to the next sequential output of the IN.

The fourth and last control mode being investigated, Closed Loop Non-Sequential, also uses the Event Request Link to indicate whether a destination processor is ready. The Trigger System Interface assigns events of specific trigger types to the next ready processor interested in that particular type of event, and again does not generate a trigger accept if there is no available processor.

The principle distinction between open and closed loop control is the point in the system where events are discarded when all processors are busy. In closed loop mode, events can be discarded at the front-end by not issuing a trigger accept or they can be redirected to a channel with a free processor. In open-loop mode, the data is transmitted but is not written to a processor.

Behavioral Modeling & Simulations Of The Architecture

The purpose of modelling and simulating the switch-based Scalable Parallel Open Architecture Data Acquisition System is to provide a learning vehicle whereby the system designers can experiment with different architectures and control mechanisms to enable them to better understand the design. This better understanding will simplify decisions such as which operation mode provides for highest throughput, what extra electronics and software should be implemented to more efficiently diagnose failures and fix problems, etc. Modeling and system simulations assist system designers in determining throughput for different configurations, identifying potential bottlenecks, interfacing to "physics data" simulations, identifying busiest channels, selecting proper buffer sizes, determining the number of processors and processing power required, determining data rates, etc.

With the ever-increasing complexity of detectors and their associated data acquisition systems, it is important to bring together a set of tools to enable system designers, both hardware and software, to understand the whole system including the behavioral aspects and the interaction of different functional units within the system. For complex systems, human intuition is inadequate since there are simply too many variables for system designers to begin to predict how varying any subset of them affects the total system. On the other hand, exact analysis, even to the extent of investing in disposable hardware prototypes, is much too time consuming and costly. Simulation bridges the gap between physical intuition and exact analysis by providing a learning vehicle in which the affects of varying many parameters can be analyzed and understood. In this way much time can be saved in the design process and one has significantly increased the probability of understanding not only the system as a whole but also the interaction of different sub-systems.

The following is a partial list of simulations which are being undertaken as this data acquisition system architecture is being developed. Simulations are divided into normal system operation simulations (i.e., no errors) and system simulations with errors. Effects of variations in the number of detector channels, event size, event distribution, front-end buffer size, number of data links, data link transfer rate, packet lengths, routing algorithms, number of processors, average processing time, processing time distributions,

processor and other system buffer sizes, event request delays, etc. will be investigated. Architecture improvements resulting from these simulations will be made. The effects of intentionally-induced errors in the system such as data errors, header errors, routing errors, buffer overflows, source failures, processor failures, switch failures, etc. will then be investigated. Architecture improvements resulting from the diagnostic simulations will then be made.

Monte Carlo data will be used in both simulations of the architecture and actual tests of the prototype system under development. A report of all simulations and hardware tests should be completed early in 1991.

Simulations are being performed using a Solbourne [6] UNIX workstation (SUN-4 compatible) running the DataViews [33] real-time graphical interface package. Links to an expert system (Nexpert [34]) for diagnostics are also being investigated. The architecture will be extensively modelled and simulated using Verilog-XL [35]. The system will have the ability to switch between simulations and the prototype hardware from a common user interface. This development technique should result in a substantial part of the runtime software needed to control an experiment using this architecture being written and tested as part of the simulations and testing of the prototype system.

System Design Methodology

From the outset of the project the goal has been to provide an integrated systems engineering environment in which hardware and software development can proceed in parallel and actually complement one another. To achieve this, it was first necessary to bring together a set of tools to not only allow extensive exploration of all aspects of the design, but also provide building blocks that would encourage the close interaction of software and hardware engineers. This approach has had the very positive advantage that valuable information is constantly being communicated between hardware and software groups during the development process. The powerful tools which were set in place included a Computer Hardware Description Language (CHDL) and simulator, a high-speed graphics package and a knowledge-based expert inference system, all running on a very powerful work station. Although each of these is very useful when used alone, when they are combined with appropriate linking software the effects are even more powerful and versatile. For example, in order to configure, download, monitor and diagnose the "model" of the data acquisition system, a user interface is being developed which best accommodates these functions. The requirements of this interface are identical to those of the actual physics experiment. If the model is an accurate representation of the actual system, then everything that a user would like to do to his system, he would also like to do to his model. Therefore, as the model is developed, the actual software used to run the experiment is also being developed in parallel in an integrated fashion.

Another example of integrated systems engineering is the development of system diagnostics and their integration into the hardware designs during the simulation process. Good systems diagnostics are crucial to minimizing downtime in a running experiment. In order to diagnose something, it helps to understand it. Before any hardware is actually built, diagnostic strategies are evolving and being tested on the "model". At the same time, one should not require hardware engineers to learn the syntax of a rule-based expert system, but one can choose a medium, such as decision trees, as the common base for storing problem-solving knowledge. It is fairly trivial for hardware engineers to represent their problem-solving knowledge in the form of decision trees, just as it is fairly trivial for programmers to translate from decision trees

to "rules" in a knowledge-based or expert system. When the programmer tests these rules, he interacts with the model and the hardware engineer to verify their operation and effect.

In the Scalable Parallel Open Architecture Data Acquisition System at Fermilab, we have already linked together the CHDL package, graphics package and knowledge-based package such that the user is now presented with an elegant "windowed" user runtime interface, in which he can select either simulated or real data taking. With the simulation mode, the user can start and stop runs, inject faults and observe their effect, and invoke diagnostic procedures to find the problem. The current technology is such that just "clicking" on a window invokes another process (e.g., simulation task), and causes rules to "fire" in the expert system, which in turn cause new "windows" or new "viewgraphs" to appear in the user runtime interface. This interface can be very conducive to narrowing down a problem, or guiding a technician or operator through problems.

SUMMARY

In the early 1980s, when data acquisition systems for many of the current generation of high-energy physics experiments were designed, bandwidths greater than ten Megabytes/second were not economically practical. The detector electronics could not generate data, and high-level processors, if they existed, could not process data at those rates. Recent improvements in technology have allowed almost a thousand-fold increase in throughput for virtually every component of data acquisition systems. VLSI front-end logic now supports synchronous readout and triggering at detector interaction rates. Processors are now typically 100 times faster and 1000 times more cost effective (compared to the original PDP-11 class machines), and can be expected to improve by another factor of ten before the end of this decade. While the inherent bandwidth of copper cable has not increased, parallel switching techniques and high-speed serial interconnects based on fiber-optic technology now make data transmission and event building at rates of 1-10 Gigabytes/second practical.

Although the performance of these systems has increased by several orders of magnitude, they are still expensive. Wherever possible, a common architecture which can be used in many different experiments and for many different triggers within an experiment is strongly preferred. The low-level triggers and much of the detector electronics are assumed to be system dependent, but beyond this point the data acquisition system should be designed for general-purpose use. General-purpose architectures do not preclude the use of special-purpose processors, which may still be more cost-effective in many cases. If possible however, both types of processing should be interchangeable.

A major goal in developing a very high-bandwidth event builder is to reduce the need for fast inline processing. After an event is fully assembled, the "real-time" restrictions on processing throughput and time-ordering of events are mostly eliminated. Trigger decisions can be more reliable when a processor has access to all of the event data, without serious processing time limitations and with the ability to easily modify the trigger algorithms. For this reason, squeezing the highest possible rejection factor from the low-level trigger logic is not always the best approach.

ACKNOWLEDGEMENTS

The authors wish to express their appreciation to members of the Data Acquisition Electronics Department of the Computing Division and Gustavo Cancelo of the Physics Department at Fermilab for their efforts on the project described in the paper, and especially to Carl Swoboda and Hector Gonzalez for their contributions to the paper.

REFERENCES

- 1 Bhuyan,L.N.et al,Performance of Multiprocessor Interconnection Networks, Computer, 1989.
- 2 Modular Instrumentation and Digital Interface System (CAMAC) ANSI/IEE Std. 583-1982.
- 3 IEEE Standard FASTBUS Modular High-Speed Data Acquisition and Control System, ANSI/IEE Std. 960-1986.
- 4 VMEbus Specification Manual, Motorola, Inc.,1985.
- 5 Silicon Graphics Inc., Mountain View, California.
- 6 Solbourne Computer, Inc., Longmont, Colorado.
- 7 Barsotti,E.J. et al, Nuclear Instruments and Methods in Physics Research A629,82-92, 1988.
- 8 Digital Equipment Corporation, Maynard MA.
- 9 Multibus II Bus Architecture Specification Handbook, Intel Inc., 1984.
- 10 Ender,C., et al ,Multiprocessor Data Acquisition System for High Event Rates at the Heidelberg/Darmstadt Crystal Ball, IEEE Transactions on Nuclear Science,Vol.36,No.5,1989.
- 11 Andrews,F., et al, No. 5 ESS - Overview, ISS 81, Vol. 3, 1-6.
- 12 Huang, A., et al, Starlite: A Wideband Digital Switch, GLOBECOM '84 Proceedings , 1984.
- 13 Anthes,G., Intel Lands DARPA Super Award, Federal Computer Week, Vol.3,No. 15, 1989.
- 14 Tucker, L., Architectures and Applications of the Connection Machine, Computer, August 1988, Vol. 21, No. 8, p. 26-38.
- 15 Meiko Ltd., UK.
- 16 Fiber Distributed Data Interface (FDDI), Draft Proposed National Standard, FDDI Token Ring Media Access Control (MAC), ANCS X3T9.5, Feb. 28, 1986.
- 17 HPPI (HSC) American National Standard X3T9.3.
- 18 Scalable Coherent Interface (SCI) Proposed Standard IEEE P1596.
- 19 UltraNetwork Technologies,Inc., San Jose, California.
- 20 Adams,G.B.III et al, Fault-Tolerant Multistage Interconnection Networks, Computer, June 1987, pp. 14-27.
- 21 Siegel, Howard Jay, et al, Using the Multistage Cube Network Topology in Parallel Supercomputers, Proceedings of the IEEE, Vol. 77, No. 12, December 1989, p. 1932-1953.
- 22 Karol, Mark J. et al, Input Versus Output Queueing on a Space-Division Packet Switch, IEEE Transactions On Communications, Vol. Com-35, No. 12, December 1987, pages 1347-1356.
- 23 Hughes Aircraft Corporation, Irvine, California.
- 24 Advanced Micro Devices, Sunnyvale, California.; Transparent Asynchronous Transmitter/Receiver Interface (TAXI) Technical Manual Preliminary Rev. 1.1
- 25 Gazelle Microcircuits, Inc., Santa Clara, California, Company Publication regarding Gazelle Hot Rod set of integrated circuits.
- 26 Berra, P.Bruce., Optics and Supercomputing, Proceedings of the IEEE, Vol. 77, No. 12, December 1989, p. 1797.
- 27 Hinton, H.Scott, Architectural Considerations for Photonic Switching Networks, IEEE Journal on Selected Areas in Communications, Vol. 6, No. 7, August 1988, p. 1209.
- 28 Murdocca,M. et al, Optical Design of a Digital Switch, Applied Optics, Vol. 28, No. 13, July 1989, p. 2505.
- 29 Gelsinger,P.,et al.,Microprocessors - circa 2000, IEEE Spectrum, 43-47,October,1989.

- 30 Barsotti, E.J., et al, A Proposed Scalable Parallel Open Architecture Data Acquisition System for Low to High Rate Experiments, Test Beams, and All SSC Detectors, IEEE Transactions in Nuclear Science, to be published in the June, 1990 issue.
- 31 Bowden, M., et al, A High-Throughput Data Acquisition Architecture Based on Serial Interconnects, IEEE Transactions on Nuclear Science, Vol.36, No.1, February 1989, p 760-764.
- 32 Briley, B., Introduction To Telephone Switching, p. 71-77, Addison-Wesley, 1983.
- 33 DataViews, V.I. Corporation, Amherst, Massachusetts.
- 34 Nexpert, Neuron Data, Inc., Palo Alto, California.
- 35 VERILOG-XL, Cadence (Gateway), Lowell, Massachusetts.