Presented at IEEE 1983 Nuclear
Science Symposium, San Francisco,
Calif., Oct. 19-21, 1983.

BNL--33736

DE84 002579

A MEMORY INTENSIVE FUNCTIONAL ARCHITECTURE
FOR DISTRIBUTED COMPUTER CONTROL SYSTEMS*

D. G. Dimmler

Brookhaven National Laboratory
Upton, New York 11973

October 1983

* This research was supported by the U. S. Department of Energy:
Contract No. DE-AC02-76CH00016.

By acceptance of this article, the publisher and/or recipient acknowledges
the U. S. Government's right to retain a nonexclusive, royalty-free license
in and to any copyright covering this paper.

A MEMORY INTENSIVE FUNCTIONAL ARCHITECTURE
FOR DISTRIBUTED COMPUTER CONTROL SYSTEMS*

D. G. Dimmler

Brookhaven National Laboratory
Upton, New York 11973

NOTICE
PORTIONS OF THIS REPORT ARE ILLEGIBLE.
It has been reproduced from the best available copy to permit the broadest possible availability.

Abstract

A memory-intensive functional architecture for distributed data-acquisition, monitoring, and control systems with large numbers of nodes has been conceptually developed and applied in several large-scale and some smaller systems. This discussion concentrates on (a) the basic architecture; (b) recent expansions of the architecture which now become feasible in view of the rapidly developing component technologies in microprocessors and functional large-scale integration circuits; (c) implementation of some key hardware and software structures and one system implementation which is a system for performing control and data acquisition of a neutron spectrometer at the Brookhaven High Flux Beam Reactor. The spectrometer is equipped with a large-area position-sensitive neutron detector.

Background

The increasing processing power and decreasing cost of microcomputers and large-scale integrated circuits induced in the past several years widespread interest in the development and the implementation of distributed data acquisition and control systems where many "intelligent units" are geographically and logically distributed, are interconnected in one way or another and are operating concurrently. Host-based distributed systems are most popular at this time. There an attempt is made to overcome the often serious overload and functional interference problems of traditional single-processor computer systems. Front-end computers, array processors, file computers and other specialized units are attached and take over some of the processing load. They also provide protection against undesirable functional interference. The central computer, now called a host or master, is often left only with supervising and controlling the traffic between the connected units. The size, functional variety and the effective combined processing power of a host-based distributed system is limited by the capabilities of the host computer in controlling and handling such traffic and by the cost and complexity of the interconnecting hardware.

The distributed function architecture as described in the following avoids such limitations. Processing units, here called nodes, are placed along data and control paths geographically as close as possible to the source or destination of the data or the control actions. Many specialized nodes rather than a few general-purpose nodes are typically used in order to keep specific node control software and hardware interconnection with the paths as simple as possible. The traditional task of "interfacing" a computer with a process or with experiment electronics is largely eliminated because the nodes are part of the electronics. Many nodes in the system act as their own masters requesting services which are offered from other nodes in the system. Because there is no central host which supervises the traffic, such a system does not have above mentioned structural limitations for

expansion as far as size, functional variety and combined processing power is concerned.

Concept

A system implemented in the distributed function architecture is geographically and logically organized along the paths of the data and control information of the application to be implemented. These paths serve as the focal point in the description and the layout of the system. Numerous nodes, each equivalent to a "computer" in the traditional terminology, are placed and interconnected along these paths. More detail on the basic concept is given elsewhere.¹⁻⁴

Modules

For practical and economic reasons the LSI-11 family of microcomputer modules of the Digital Equipment Corporation has been used. These modules connect to the Q-bus. A large variety of hardware modules and application software packages are commercially available for this line of microcomputers. Work is underway to incorporate Q-bus based as well as VME-bus based 68000 microcomputers.

Nodes, Clusters, and Cluster Memory

In general, nodes perform one function or a set of compatible functions in order to keep the node complexity low. A typical system, therefore, consists of many nodes. Several nodes can be interconnected to a cluster. As shown in Fig. 1, each horizontal line represents a Q-bus with its own LSI-11, channels, device electronics and local memories. Each Q-bus is

Fig. 1. Cluster Organization

connected via an access port to the M-bus shown as a vertical line. The M-bus consists of an extended UNIBUS where the address bits 22 to 27 have been added. Thus up to 270 million bytes of commercially

* This research was supported by the U. S. Department of Energy: Contract No. DE-AC02-76CH00016.

available UNIBUS memory modules can operate as a cluster memory which is accessible from all the nodes in the cluster. Figure 2 shows a photograph of a typical

Fig. 2. Typical cluster node

node where the Q-bus is placed on the two right-hand sections of the backplane and the M-bus is placed on the left hand section. The M-bus sections of several interconnected backplanes, then, form the M-bus. Figure 3 shows a photograph of a typical cluster configuration. Up to 5 nodes and several million bytes of cluster memory fit into a typical 19-inch wide drawer.

Fig. 3 Typical cluster

Access control to the cluster memory is accomplished using a round-robin token passing algorithm. An access control module offers access to a node for 80 nanoseconds, via two coaxial cables as shown in Fig. 2. The node has the option to execute one memory cycle after which the token is given to the next node in line. This algorithm appears advantageous as compared to priority algorithms in that a high priority node cannot lock the cluster memory by claiming all accesses. The access offer time will be reduced to 20 nanoseconds in a new design presently under development. The new design will also include a dynamically configurable list which allows for access priority assignments. Nodes with higher priority are listed more often than nodes with a lower priority. Presently up to eight nodes may participate in a cluster. The new design will allow for a much larger number of participating nodes.

Q-Bus Address Space

The Q-bus address space is partitioned as shown in Fig. 4. Bits 00 to 12 of the addresses falling into the 4096 word window between 140000(24K) and 160000(28K) are routed to the M-bus access port. There the bits 13 to 27, which had been previously stored in a page register, are merged and the result is used as the physical address in the M-bus. Each 4096 word page can be protected against accidental write accesses using a protect bit for each page in

two 2716 erasable read-only memories. The read-only memory is located in the access port module.

Fig. 4. Address space partitions in a node

Connections Between Clusters

Closely Coupled Clusters

Clusters which are geographically located closer than 15 feet and are functionally compatible may be closely coupled by interconnecting the M-buses of the clusters. Communication is accomplished across the cluster memories which are then accessible by all nodes participating in the closely coupled arrangement.

Long-Distance Communication

Clusters which are geographically separated more than 15 feet up to 12000 feet or are assigned functions which are not easily compatible, are connected with the long-distance communication network. In this case one node in the cluster is assigned the function of the communication terminal which controls and performs the long-distance communication for all other nodes in the cluster. The topography of this network is somewhat different than in traditional local-area networks, most of which are based on multitapped loops, rings, or lines. This network is intended to be used within control, monitoring and data acquisition systems in experimental environments whose requirements change often and sometimes in unpredictable ways. It has been considered vital that such a network can be easily reconfigured, expanded, partitioned into several networks and merged as the needs arise. Very often it is also necessary to broadcast information from one node to many others in different clusters. Also multiple, redundant paths may be required between communicating nodes for reasons of increased reliability and interference protection. For above mentioned reasons, and also for reasons of functional simplicity, a meshed interconnection scheme has been chosen.

A configuration example of a meshed network is given in Fig. 5. At any given time each node in the network assumes the role of either an originator of a command or a server of a command. A command includes the data to be communicated and instructions on what to do. An originating node, in the example of Fig. 5 the control node A, submits a command via a PUT transaction, as outlined in Fig. 6, to a communication switch. The command contains its own unique 32 bit identification number, the identification number of the command server or, in the case of broadcasted commands, a list of those numbers. Eight bits of the identification number are reserved for a unique network address. Sixteen bits are reserved for a cluster address which is unique within a network and 8 bits are reserved for a node address which is unique within a cluster. The switch routes and queues the transaction into the wait buffer set aside for the receiving server node as shown in Fig. 6. Concurrently all nodes which are open for service requests will interrogate their wait buffers in the switch or in several

Fig. 5. Network with multiple paths

Fig. 6. Communication between originator and server via the communication switch

switches for work to be performed. The command, once arrived, will be picked up via a GET transaction by the server and will be processed. In the case of a WRITE command with an acknowledge or a READ command, the return information is transferred the same way only in the opposite direction. In that case the originating node assumes, during the return phase of the command, the role of a server.

The communication hardware consists of a point-to-point link between each participating cluster and each switch. The link consists of one coaxial cable for each of the two directions. An 8.5 MHz carrier is frequency modulated with a 1 MHz data stream. This arrangement tends to avoid distortion problems as often experienced in base band communication links and the often experienced electronic complexities at the tap points of broad band communication schemes. The receiver, in fact, consists of a single RCA CA 3189E television receiver chip. The transmission is accomplished by a three-stage voltage controlled multivibrator chip. As can be easily seen in Fig. 5 a cluster can be connected to several switches allowing for broadcasting and multiple path transmission. A cluster can also be a member of two otherwise independent networks. The arrangement will act as two independent networks if in Fig. 5 the cross connections are not used.

Communication Protocol

The communication protocols between closely coupled clusters and between nodes within a cluster are usually specific to the function to be performed. Only general guidelines are established here. The control flags and data are communicated through agreed upon pages in the cluster memory. It was felt that a general-purpose function-independent communication protocol would here have the tendency to substantially increase functionally unrelated overhead causing a decrease in speed and an increase in logistic complexity. Clusters, on the other hand, which are coupled over the long-distance network, follow a very strict function-independent protocol as has been briefly outlined before.

Network Complexity

As a general rule, networks in experimental environments tend to show their optimal usefulness and reliability if they include only features which are: a) well understood by the users, b) are designed functionally complete in the sense that there is a well understood response to every conceivable error condition and c) are used with some regularity. Seldom used or poorly designed features tend to increase confusion and tend to impact adversely the usefulness of the entire network. Therefore, systems ought to be built up incrementally, starting with simple configurations and expanding to more complex schemes if: a) a demonstrated need exists and b) the functional completeness of the new features can be guaranteed. The meshed communication network scheme allows for such a strategy.

A Control System for a Neutron Spectrometer

A system which controls all the motions in a neutron spectrometer and takes data from an associated 20 by 20 centimeter position sensitive detector and other data sources has been implemented in this architecture and has been installed at the Brookhaven High Flux Beam Reactor. Five clusters are participating in the system as shown in Fig. 7. Three clusters are functioning as work stations. Two clusters are closely coupled; they function as the communication switch and the data storage module.

Experimental Floor Work Station

The experimental floor work station, which is located very close to the spectrometer, includes six nodes. Three of these nodes, the spectrometer control node, the analysis computer node and the graphic display node each operate FORTRAN application programs much in the same way as three traditional computers. Two nodes, the encoder service node and the motor/scaler service operate invisibly to the user of the system as input/output function and experiment control nodes. Fig. 8 shows a more detailed layout of these two nodes. Equipment which is in traditional systems labeled as "experiment interface and device electronics" is here inserted into the appropriate Q-bus as an integral part of the node. Included are the necessary scalars, display monitor control units, motor control units, motor translators, encoder input electronics, as well as most of the signal conditioning and the data input from the position sensitive detector for a maximum input rate of 400,000 events per second. Most of these modules have been designed in-house making elaborate use of commercially available large-scale integrated chips. Fig. 9 shows, for instance, a module which controls independently the motion of eight stepping motors allowing for all necessary functions including a variety of dynamically selectable ramping parameters, 256 dynamically selectable speed parameters

POSITION-SENSITIVE DETECTOR SYSTEM AT HIGH FLUX BEAM REACTOR NETWORK 21

Fig. 7. Neutron Spectrometer Control System

Fig. 8. Detailed layout of the motor/scaler service node and the encoder service node

ranging from 15 steps per second to 5000 steps per second, recognition of two soft and two hard limit switches for each axis as well as an automatic search of the zero position of each axis. The stepping motor module delivers to the motor translator module, a photograph of which is shown in Fig. 10, the three-, four- or five-phase pulse trains necessary for each axis depending on the type of motors being driven. The module shown in Fig. 10 houses four translators switching motor currents of up to 10 amperes. The power for the motors is taken from a common 15 volt, 750 watt power supply. There is no additional interface

Fig. 9 Photograph of a module controlling eight stepping motors

or driving electronics located between this node and the motors. Incremental encoders track dynamically the position of each axis. Three lines for each encoder are brought into the encoder service node via the signal conditioning module and the encoder interrupt module at the encoder service node bus as shown in Fig. 8. The three lines carry pulses for clockwise and counter-clockwise movements as well as for the pulse which is transmitted once per revolution at the instant when the encoder moves over the index point. In the worst case the node must service one interrupt pulse each 100 microseconds. In order to avoid pulse

Fig. 10. Photograph of a motor translator module

losses each line includes a 128 bit deep FIFO buffer for derandomizing. In addition, the node is dedicated exclusively for this interrupt service. Each motor is described with two device descriptor blocks of 128 sixteen-bit words each, a constant descriptor block and a variable descriptor block. The constant descriptor block includes amongst other parameters the name of the axis, the location, the angular limits of motion, the speed chosen, the acceleration and deceleration angles and the various conversion factors necessary to translate between the floating point angle descriptors, the number of pulses per revolution of the motor and of the associated encoder, as well as the display representations of the motor target and the present encoder positions. The variable descriptor block contains the present angular position of the appropriate axis, the last target position which had been requested and a variety of other variable parameters. The descriptor blocks are located in the cluster memory of the experimental floor work station. All nodes in the clusters have read access to the blocks. The motor/scaler service node and the encoder service node have, in addition, write access to both descriptor blocks. Other devices, such as scalars and temperature controllers, etc. are described in a similar way using appropriate descriptor blocks in the cluster memory. The status and position of the motors and other devices of the neutron spectrometer are displayed using a series of raster scan monitors. Up to three different images can be displayed concurrently. Fig. 11 shows a typical set of images.

Fig. 11. Photograph of three display images at the experimental floor work station

Storage Module and Communication Switch

All data files as well as the load modules of all programs used in the system are stored in a storage module cluster which is directly accessible by every node in the system. The cluster is closely coupled to the communication switch in order to achieve optimal communication speed.

Peripheral Equipment Work Station

This cluster operates the magnetic tape unit and the printer. Its node assumes the role of a server node. Experimental floors are often an unsuitable place for temperature and dust sensitive computer disk units and peripherals. Therefore the peripheral equipment cluster and the storage module cluster are housed in an enclosed room a few hundred feet away from the experiment, taking advantage of the meshed long-distance communication network.

Software Development Work Station

Functionally and geographically distributed systems have the tendency to require large quantities of error-free, functionally complete and often highly function-specific application and system software. The present state-of-the-art technology in software development tends to be: a) tedious, b) very time-consuming, c) very susceptible to human error and d) often confusing. Limitations in software development may turn out to become a considerable bottleneck in the application of modern real-time systems. An attempt has been made to improve this situation by moving towards automation of the often considerable procedural portion of the software development process. A software development workstation including three nodes has been developed and interconnected with the network. The object manager node, as shown in Fig. 7, operates an automatic command and name generation program, feeding an RSX11 operating system. The source manager node operates a language specific text editor. Five raster-scan display monitors are connected to the source manager. As shown in a photograph in Fig. 12, three monitors display directory information of a text

Fig. 12. Photograph of a software editing workstation

module which is in the process to be edited as well as information for other text modules to allow for comparison; one monitor contains the software text module to be edited and the fifth monitor allows for visual comparison with another text images. Updating of the directories is done during and after the editing session automatically without user attention. After a software module has been edited it is shipped via the cluster memory together with all necessary directory information to the object manager. The command and name generation program generates a complete set of

commands and names necessary for compiling and insertion of the module into the libraries and for all other associated functions. Also the linking and managing of complex overlay systems which may include up to 128 overlay tasks are directed by the object manager without user attention. Thus, software changes in very elaborate program systems can be easily made. Load modules are, after linking, transferred via the long-distance communication network to the storage module where they are recorded in the task library module. They are then available for loading by the appropriate node in the network.

Other Systems and Future Plans

Thirty nodes with an LSI 11 processor each, several million words of random-access memory and 300 megawords of disk space have been used in a system⁵ which monitors a series of animal inhalation chambers at the Brookhaven Inhalation Research Facility. The system, which is in routine operation since several years, also collects data from long-term animal exposure experiments, distributes and tracks alarm conditions in and around the chambers and informs on a continuous basis to facility personnel the status of the various ongoing experiments. The system which is in 24-hour per day operation throughout the year, has a near-perfect operational record.

Several projects are underway using this architecture for systems related to high-speed data acquisition and manipulation in neutron and x-ray proton crystallography at the Brookhaven High Flux Beam Reactor and at the National Synchrotron Light Source. A high-speed data collection station will allow for connection of several position-sensitive neutron and x-ray detectors. Each channel may have a maximum data rate of 5×10^5 events per second and allows for time-slice operation. The time-slice resolution will range from 5 microseconds to several hours. A segmented M-bus is under development which allows for several concurrent data input streams into the cluster memory. The combined total event rate, thus, may be extended to several million events per second.

Acknowledgements

I am deeply indebted to Mr. S. Rankowitz who developed most of the custom-designed electronic hardware modules. Ms. M. A. Kelley designed much of the FORTRAN software. Mr. J. Gannon's services in testing the modules and subassemblies and in integrating the various systems are greatly appreciated. I would like to express my appreciation to Messrs Rankowitz and Potter for extensive discussions and advice on the hardware aspects of the architecture and the implemented systems.

References

1. D. G. Dimmler and W. H. Hardy, II; A Shared Random Access Memory Resource for Multiprocessor Real-Time Systems. BNL 21884. IEEE Trans. Nuc. Sci. NS-24, No. 1, February (1977) 469-475.
2. D. G. Dimmler; Computer Networks in Future Accelerator Control Systems. BNL 23433. IEEE Trans. Nuc. Sci., NS-25, No. 2, April (1978) 974-988.
3. D. Gerd Dimmler; Results of an R&D Effort on the ISABELLE Control System. BNL 25528. May (1979).

4. D. G. Dimmler; A Memory-Intensive Functional Architecture of Distributed Computer Control Systems. BNL 31199. Proc. DCCS-82, 4th IFAC Workshop on Distributed Computer Control Systems, Tallinn, USSR, 24-26 May (1982).
5. D. G. Dimmler and D. M. Bernstein; A Distributed Microprocessor Network for Control, Monitoring and Data Handling at an Inhalation Toxicology Facility. 1982 Society of Toxicology Annual Conf., Boston, MA, 22-26 February (1982).

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.