JAMES G. PALMER

COMPUTER SYSTEM ARCHITECTURES FOR FUTURE NAVY TACTICAL SYSTEMS

The Navy relies on real-time embedded computers for the operation of radar, missile, and decision support systems. Dramatic advances in computer technology offer possible solutions to the unique problems of future tactical systems. Candidate architectures based on the technology options are examined and related to selected Navy projects.

INTRODUCTION

Since the introduction of computers as an integral part of weapons systems, the performance of many Navy systems has depended on the processing capacities of their embedded computers. Because of constraints on equipment space, weight, and cost, many Navy embedded systems have been tailored to initial requirements, only to find that long-term growth required costly redesign and replacement of computer equipment. Computer technology has progressed rapidly in recent years and consideration of new Navy embedded computer system architectures is warranted. Pressures are growing for substantially increased processing power to solve critical problems in real time, such as engagement-level sensor integration for the NATO Anti-Air Warfare (AAW) System, the Cooperative Engagement Program for the Battle Group AAW Coordination Program, and numerous others.

The military shipboard environment also poses a major challenge to the design of future high-performance computer equipment. The severe shock, vibration, temperature, and humidity that can occur during battle would rapidly disable most commercial computers; military equipment is built to environmental specifications to overcome these problems. No equipment will survive localized battle damage or a serious fire, however, and most ships are not designed with sufficient redundancy to maintain system operation when local damage occurs to their computer systems. The redundancy and flexible reconfigurability necessary to achieve survivability could now be provided by improved data distribution techniques, redundant processors, and distributed software techniques.

Planned upgrades to the Navy's standard shipboard computers offer substantial performance improvement and compatibility with existing software (although consideration of the standard computers in this article relates primarily to the current versions of these computers). Also, the Next Generation Computer Resources Project is a program to standardize several computer system technologies for fleet use. Efforts toward an open architecture for a backplane bus (i.e., local to a single chassis) and local area network (LAN) are important to future Navy systems. Similar technologies are being put to immediate use by experimental and prototype development projects such as the Automatic Identification System, the Cooperative Engagement Program, and the NATO AAW System.

The study reported here identifies some of the options in computer system technology and associated issues for Navy systems and focuses on the integration of promising technologies into computer system architectures that are appropriate for meeting future needs of the embedded computer systems. The study was originally oriented to requirements for the NATO AAW System; the need for highly flexible, adaptable architectures to accommodate multinational needs applies to a more general set of Navy problems, however.

GENERAL COMPUTER SYSTEM REQUIREMENTS AND DESIGN GOALS

Requirements for Navy embedded computer systems include system performance, physical constraints, and the need for flexibility in applications to different combat system configurations. The specific system requirements for any application are typically defined in a corresponding system specification. The following are generally important for any embedded system, however.

System Response Time. Requirements for system response time must be met, and certain mission-critical timing requirements must be satisfied under all conditions, regardless of other processing in progress. Response time requirements drive the specifications for computer processing power, interface throughput, and appropriate control to support quick response without operator intervention. Performance evaluation must be made on the basis of threat scenarios that exercise the full range of required response times and capacities.

Operational Availability/Survivability. The mission-critical nature of many embedded applications means that the computer system must be able to recover automatically from the failure of any element or of any interconnection of the computer system. The computer system design should provide for the maximum feasible availability and survivability. To offer survivability from local damage, it is essential that multiple processors and associated interconnections be physically distributed and have alternate interconnection paths.

Space and Weight. Designs must accommodate deployment on ships that have small space and weight margins or in even tighter constraints, such as on aircraft or placement in existing shipboard enclosures.

Design Goals. In addition to the above critical requirements, the following system-level design constraints are often imposed on the computer system:

1. Navy directives require that all future applications software be written in Ada, the DoD standard programming language.
2. The design should be adaptable to system variations and should maximize reuse of software among these variations.
3. The design must provide for substantial growth (e.g., a factor of at least two to allow for enhancements during development and major additions through the system life cycle).
4. The design must support flexible control without requiring a unique program code for each tactical situation.

SOFTWARE ARCHITECTURE CONSIDERATIONS

In developing a computer system architecture, the requirements and design constraints of the software impose critical, but often overlooked, requirements on the computer equipment architecture. A few of the many software ramifications are presented in this section.

Programming Language and Run Time

A system's programming language can have a major impact on run-time performance, program development cost, reuse of existing software, and maintainability over the system's life cycle. Although the use of assembly language can often give the best run-time performance, the development and maintenance processes would be so inefficient that assembly language would be used only as a last resort for small portions of programs that cannot be implemented otherwise. Very few high-order languages provide adequate code execution efficiency for real-time applications. For the Navy standard computers, the Navy-unique CMS-2 language has been used for real-time programs. It is the Navy's intent (and a DoD requirement) that Ada fulfill real-time computing needs for future mission-critical systems. The Ada Language System/Navy is now available for the Navy's AN/UYK-43 and AN/UYK-44 computers, with a multiprocessing version due in 1990. Assessment of the run-time efficiency of UYK-43 and UYK-44 code is in progress. For certain minicomputers and microcomputers, rapid progress in the implementation of various Ada compilers has resulted in acceptable real-time products for many applications. For very time-critical real-time applications, some Ada run-time performance problems remain. Certain run-time features are optional or are stated ambiguously in the Ada specification, which hinders progress in achieving critical run-time goals. For effective Navy use of Ada, the optional features must be supported, and the conventions for the unspecified tasking characteristics should be defined consistently. Such improvements are under consideration for revision to the Ada standard.

System Adaptability

One of the architectural goals for many computer systems is to provide adaptability to subsystem variations without major impact on the bulk of the computer hardware or software. System-unique features that can be processed in a "front-end" processor can avoid impact on other software throughout the system. A design ap-

COMPUTER PROCESSING LOAD ANALYSIS

The required computer processing load is a primary factor in determining the computer system architecture for a system. The processing load is the instruction execution rate required by the computer system to perform all required functions within specified system time constraints. It is difficult to determine accurately the required processing load without implementing the required program in the target language on the selected target computer system; however, it becomes necessary to estimate processing requirements at various stages of system development.

Processing load requirements of programs or subprograms can be expressed in various ways, such as millions of instructions per second (MIPS), floating-point operations per second (FLOPS), or as a percentage of a known computer CPU capability. Use of FLOPS is relevant for programs that are primarily mathematical. Use of millions of instructions per second can be misleading because of variances in machine instruction efficiencies and methods of instruction time measurement. The percentage of processing load of a specific CPU is a useful metric when comparing programs targeted for the same computer and language, but conversion of estimates to other target computers can introduce inaccuracies.

For real-time programs, the averaging of required processing over some time period (typically 1 s) may mask short-term response requirements in the 1 to 10 ms range. Accurate analysis therefore requires that the processing load be examined over the critical time intervals of the target system. Also, use of multiprocessor subsystems to handle a common load via task sharing must account for the limitations of load balancing that can practically be achieved.

On the basis of several existing and planned Navy tactical embedded systems (e.g., NATO AAW, Aegis 2000, and Cooperative Engagement Capability), it appears that the primary need is for computer system capacity in the range of 5 to 50 million instructions per second. There is also a need for fewer systems with computationally intensive applications to operate in the range of 50 to 200 million instructions per second, such as antisubmarine warfare sonar processing and electronic warfare/electronic warfare support measures processing. Architectures in this study will therefore concentrate on solutions that can be easily configured for the range of 5 to 50 million instructions per second, with possible incorporation of additional processing capabilities to extend performance to the 200 million instructions per second range.

proach that isolates the device-specific processing from device-independent processing can often be used to achieve maximum configuration adaptability. Figure 1 shows a software design that incorporates such layered design concepts.

Prototype designs indicate that significant benefits accrue to so-called loosely coupled systems, whereby the processing of elements is primarily via message passing, thus keeping the use of shared data to a minimum. Establishment of a controlled message dictionary to define messages within and among systems then becomes an effective means of management and process coordination.

Subprogram Partitioning

Clear partitioning of functional elements is an essential element of good system design; the same principle applies to software. Although large programs in single computers can be partitioned by design convention to achieve functional separation and well-defined interfaces, partitions are rarely maintained consistently throughout the life-cycle evolution of the program.

The use of separate processors within computer systems to create well-defined program partitions can simplify the logical as well as the developmental and control aspects of the system design. For very large systems (e.g., C^3 systems), there may be acquisition advantages to incrementally developing, testing, and deploying subsystems versus one large development. Carrying this concept too far, however, can result in fragmentation and interfacing inefficiency; therefore, optimal partitioning is a critical aspect of the computer system architecture.

It is important that the design goals related to partitioning be identified and followed consistently, allowing exceptions only when merited. The following is a set of processor partitioning guidelines:

1. Associate subprograms with clearly defined functions, grouping one or more subprograms per processor.

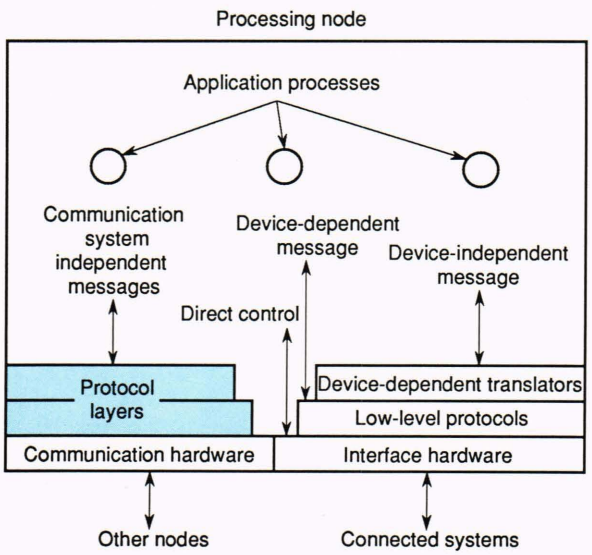


Figure 1. Software layering concepts.

2. Minimize interprocessor interface loads and response-time requirements; minimize shared data, and design for routine message-passing interface versus high-priority interrupts where possible.
3. Limit projected processing load per central processing unit (CPU) to 50%; provide for adequate performance and initial growth reserve.
4. Isolate device-dependent processing to separate processors where feasible; simplify development and maintenance of system variants.
5. Distribute local bus usage (where applicable) to accommodate peak and average loads; provide for adequate response time and growth.

ARCHITECTURE OPTIONS

Computer system architecture encompasses both equipment and software aspects of computer system design. The physical (equipment) aspects include the processor technology to be used; the processor interconnection techniques; and the configuration of selected computers, interconnection equipment, and computer system peripherals.

Computer Technology Options

Critical system requirements driving the selection of Navy embedded computers are processing speed (instructions per second), response-time constraints, interface characteristics, extensibility to meet growth needs, compatibility with the Ada language, commonality with other systems, reliability, survivability, and cost.

For Navy ships, the Navy standard UYK-43 and UYK-44 computers provide rugged, reliable service; they are supported routinely in the Navy's logistics and maintenance training programs. The UYK-43 is a 32-bit computer that can be configured as a single or dual CPU multiprocessor to provide 2.25 or 4.5 million instructions per second, respectively.[1] The UYK-44 is a smaller 16-bit minicomputer that is rated at approximately 0.9 million instructions per second.[2] Each computer also includes fast-response, multiple-channel input/output capabilities. Performance upgrades are in development for both the UYK-43 and UYK-44, which will increase the useful life of these computers. Validated Ada compilers for the UYK-43 and UYK-44 are now available for general use.

The widespread use of certain commercial minicomputers (e.g., Digital Equipment Company's VAX) has led to licensing agreements for production of militarized versions of these computers. Since the processing power of single-board microprocessors is now equivalent to many minicomputers, the introduction of vendor-unique minicomputers is not an effective approach for future Navy tactical systems.

Board-level computers based on 32-bit microprocessors such as the Motorola 68020[3] or Intel 80386 are being adopted extensively in the United States and in NATO countries for commercial, industrial, and military applications. Very-large-scale integration techniques have yielded major improvements in the performance-to-weight ratio, yielding 4 to 6 million instructions per second from current single-board microcomputers. Single-board computer performance is rapidly increasing as evi-

denced by recently announced microprocessors in the 10 million instructions per second range. Board-level computers are being militarized and embedded into combat systems (e.g., the AN/SPS-48C search radar and Mk 74 Tartar Fire Control System). With the availability of low-cost, single-board computers, the clustering of multiple boards on a high-speed bus provides even greater processing power. By carefully designing to minimize bus contention, a cluster of five microcomputer boards on a backplane bus can now achieve processing on the order of 20 million instructions per second. By adopting an industry standard backplane bus (e.g., VMEbus, Multibus II, or Futurebus) for communication within a processing cluster, the resulting open architecture will accommodate upgrades of processors as growth requires. The significant reduction in the number of replaceable card types for microprocessor-based computers potentially simplifies logistics support and maintenance over computers requiring a larger number of card types. Real-time executives and efficient high-order languages now exist for most microprocessors. Several validated Ada compilers are now available for the M68020 family, and run-time performance is approaching that of other languages commonly used for real-time systems.

Several companies have developed "super-micro" computers based on multiple-microprocessor parallel architectures. By providing a central task scheduler for typically 10 to 30 microprocessors, a multiple-instruction-stream, multiple-data-stream (MIMD) parallel architecture is achieved as exemplified by the Transputer and Convex computers. Ada compilers being developed for such machines will offer true concurrent real-time processing in the future.

For some applications, another effective form of parallel processing uses a single-instruction-stream, multiple-data-stream (SIMD) architecture (e.g., Goodyear's militarized Associative Processor). Operations on data arrays can be performed very rapidly; detailed analysis of candidate applications, however, is important to determine how the mix of parallel versus sequential operations will affect overall efficiency. MIMD and SIMD technologies can perform in the 20 to 200 million instructions per second range, which may be applicable to certain classes of problems.

Processor Interconnection Options

The method of interconnecting processors and system elements is a key aspect of computer system architecture. Significant requirements for interconnection methods are speed of data transfer (bandwidth), response time of transfer (latency), ability to reconfigure to alternate paths, susceptibility to electromagnetic interference, weight and space of cables, and commonality of protocol and equipment. Often a combination of interconnection methods is desirable within a system.

Point-to-point methods can provide high-speed, dedicated channels with guaranteed access (except in cases of failure). Weight can be minimized by using serial or fiber-optic techniques rather than parallel cables. Reconfiguration requires redundant cable paths with electronic or mechanical switching.

To provide communication within a chassis, several commercial standards exist for high-speed local (backplane) buses. A local bus provides a flexible method of interfacing board-level computers, memories, and interface cards within a single chassis, with the option to reconfigure dynamically in case of failure. Although bus speeds of 10 MB/s and higher are commonplace, care must be taken in system design to avoid overloading the bus.[4] The VMEbus and Multibus II are buses that are candidates for Navy applications. Although the VMEbus is used in more applications, Multibus II has the advantages of interprocess message handling and dynamic reconfiguration over the VMEbus. The IEEE 896.1 (Futurebus) has been selected as a future Navy bus standard; since the Futurebus standard is still evolving, however, it is not yet available for Navy use.

For system-wide (longer distance) interconnections, various LAN options are available. Communication among many users is provided by LAN's, using time or frequency division multiplexing of a common media—typically coaxial or fiber-optic cable. As the number of system interconnections increases, networks can offer advantages over point-to-point, including fewer interfaces per subsystem, less cable weight and space, flexibility for system growth, and simpler casualty/survivability reconfiguration; however, data transfer delays can be significantly larger than point-to-point interconnections. Commercially popular networks such as Ethernet may provide adequate overall bandwidth; for some real-time applications, however, they do not offer adequate responsiveness (latency) for high-priority transfers, although Ethernet performance can be significantly improved by designs such as the militarized triple Ethernet design for the Royal Netherlands Navy. Token passing bus and token passing ring designs are intended to provide responsiveness to real-time events and are becoming popular in industry (e.g., GM's Manufacturing Automation Protocol [MAP] and the Navy's SAFENET I). Furthermore, very high speeds (100 MB/s and higher) are achievable in fiber-optic-based networks such as the proposed Fiber Distributed Data Interface (FDDI) standard that is being adapted for Navy use (SAFENET II). Prototype FDDI-based interface units are being developed for military use by companies such as UNISYS, Martin Marietta, and Farranti.[5] Other networks for the Canadian, United Kingdom, and German navies are also being developed. All approaches need to be appropriately configured to provide casualty and survivability mode operation.

OPTIONS FOR COMPUTER SYSTEM CONFIGURATIONS

The development of a computer system architecture for an embedded application involves configuring the required computer(s), integrated by appropriate interconnection method(s), to support processing loads, interface requirements, internal software tasking control, casualty reconfiguration, and other system requirements as summarized earlier. To analyze a cross section of architectures, possible configurations were first represented by generic configurations distinguished by the degree of centralization versus distribution and by interconnection

methods. Variants could then be compared on the basis of alternative computers, specific interconnection systems, and casualty configuration options. Selected examples are discussed in this article.

Centralized Architecture

A highly centralized configuration that provides all processing in a single large processor or in a tightly coupled multiprocessor (e.g., dual CPU UYK-43) is shown in configuration 1A. Navy shipboard computers typically interface with 4 to 10 or more other devices such as sensor equipment, weapons equipment, other computers, operator interface equipment, and computer system peripherals (disk, tape drive, printer). In configuration 1A, all interfacing subsystems connect directly to the computer by point-to-point interconnections. For smaller applications (less processing and/or input/output capability), the central computer could be replaced by the UYK-44 or AN/AYK-14 (airborne-equivalent) computers. To provide for casualty reconfiguration with the centralized approach, fully redundant input/output and an option to switch to either of the UYK-43 CPU's and associated input/output controllers are indicated by alternate interconnection paths. This configuration provides no survivability mode in case of local battle damage.

To provide additional processing power and survivability options, a second computer is required and must be located in a separate space (configuration 1B), supplying up to 9 million instructions per second if the UYK-43 is selected. This configuration uses point-to-point (parallel or serial) channels. Backup input/output channels are again provided for all interfaces; interfaces for this option, however, require reconfiguration to the alternate computer. To extend the approach to accommodate combinations of failed computers and interface channels can require a complex array of switches or redundant connections.

This configuration can be simplified by using a local area network (LAN). One possible LAN configuration is shown in configuration 1C, using dual networks with a network bridge or some form of internal network reconfiguration for casualty options. The LAN connections require that either a network interface be built into each subsystem (as shown) or that separate network interface adapters be connected between subsystems and the network. (Configuration 1C shows a generic network with no attempt to distinguish bus versus ring topologies.) The LAN option permits flexible channel selection at each interfacing subsystem, independent of which computer is connected to the subsystem.

A similar architecture may be achieved by using upgraded UYK-43 or UYK-44 computers or militarized superminicomputers to attain better performance and growth potential.

Federated Architecture

To provide federation of subsystem control, assure timely interface servicing of connected subsystems, and relieve the processing load on the central computer, configuration 2A shows computers dedicated to subsystem processing in keeping with goals for a layered design. This approach is commonly applied for independent sensor or fire-control processing in combat systems. The UYK-44 is typically used for applications not requiring the capacity of the larger UYK-43 computer.

Configuration 2B adds redundancy for survivability. Again, providing for survivable reconfiguration options as shown in configuration 2B creates problems of logical and physical switching complexity. Consequently, design compromises that limit reconfiguration flexibility and affect computer system load balancing are often necessary. As the number of interconnected systems and subsystems increases, it is particularly beneficial to simplify interfacing and improve reconfiguration flexibility by using a LAN, as shown in configuration 2C. These configurations might use any mix of UYK-43 or UYK-44 computers, but reconfiguration is simplified if only one computer type is used.

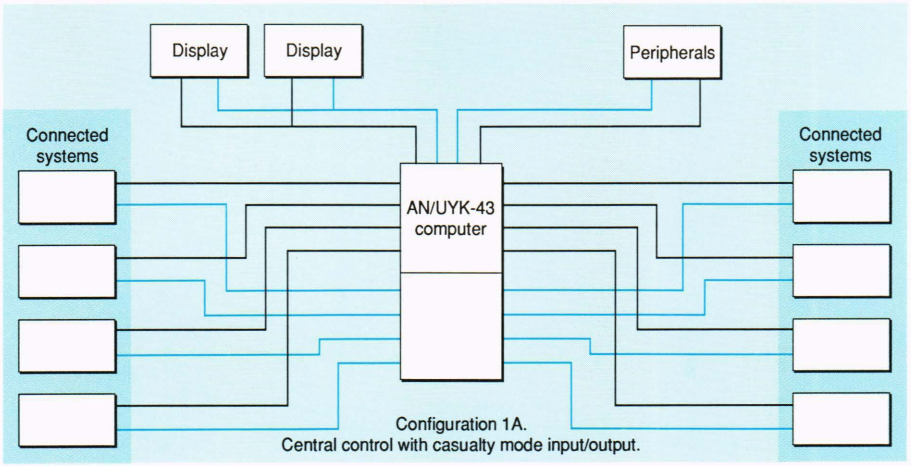
AVAILABILITY AND SURVIVABILITY OPTIONS

Achieving high system availability and survivability from local battle damage is a critical requirement for Navy systems. The following are options for achieving high system availability:

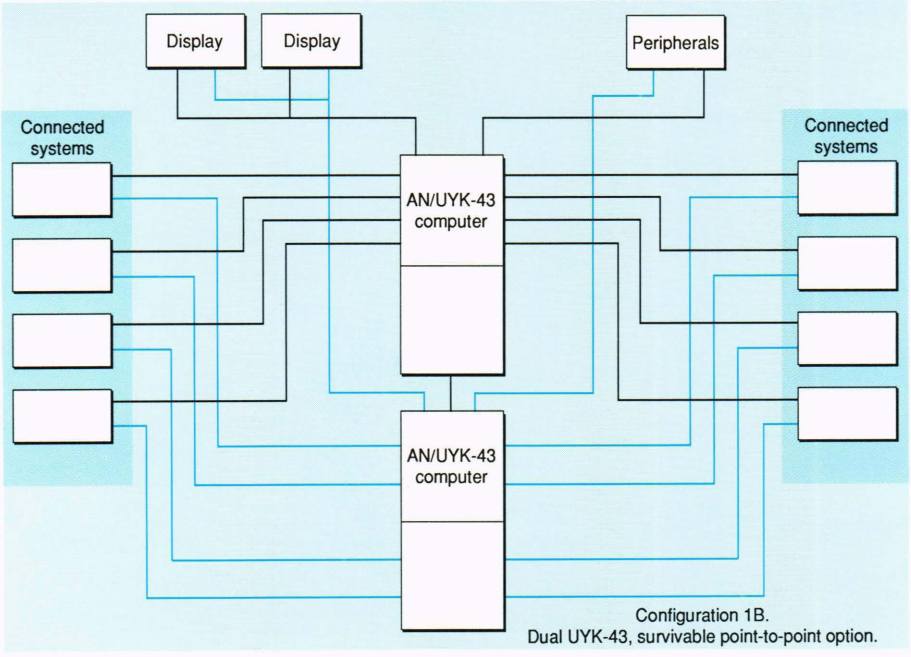
1. By designing components for extremely high reliability and low mean-time-to-repair, high availability (percent of time operational) is achieved. Extremely-high-reliability components should be a goal; however, this approach does not offer immunity to battle damage. Consequently, this option alone is not sufficient.
2. Reconfiguration of multiple computers to bypass a failed computer is a common approach to provide a fail-soft capability (e.g., the Aegis Combat System and the AN/SYS-2 Integrated Automatic Detection and Tracking System use this approach). In some cases, this requires provision of spare computer capacity so that mission-critical performance is maintained. Physical separation of processors is required to provide survivability; however, the capacity of remaining processors in the event of battle damage may be inadequate.
3. Full redundancy of all critical computers and related components permits rapid automatic reconfiguration from a failed component to a ready spare and permits reconfiguration to alternate equipment in the case of local battle damage without loss of mission-critical performance. This requires effective fault localization, a reassignment mechanism, and avoidance of single-point failure modes.

Redundant interconnection among critical subsystems is necessary to ensure connectivity if any single input/output controller or cable fails. This redundant interconnection approach was adopted with Aegis for all computer interfaces. Cable space and weight can become significant if point-to-point parallel (versus serial) cable is used for this form of redundancy.

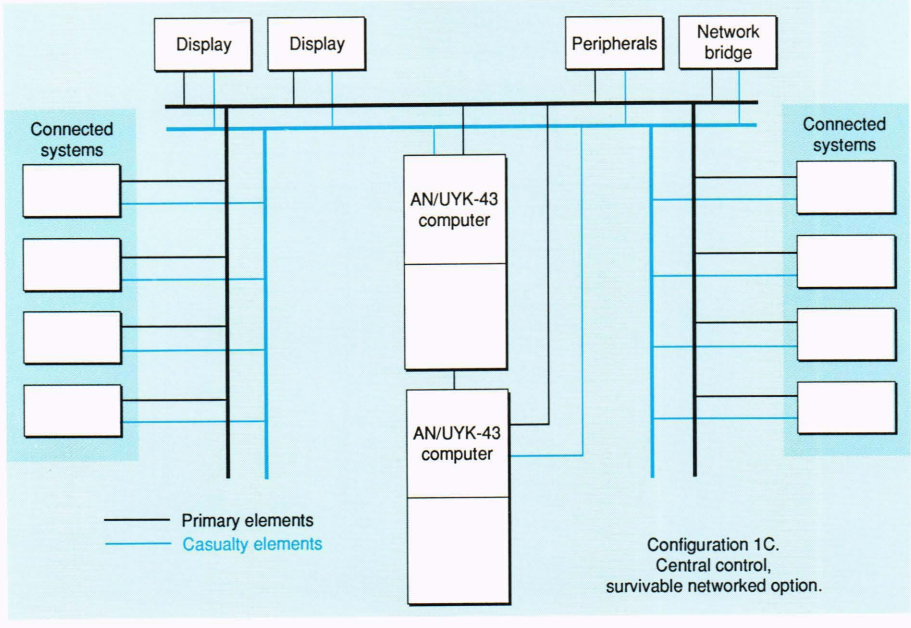
CONFIGURATION 1 OPTIONS



Configuration 1A.
Central control with casualty mode input/output.

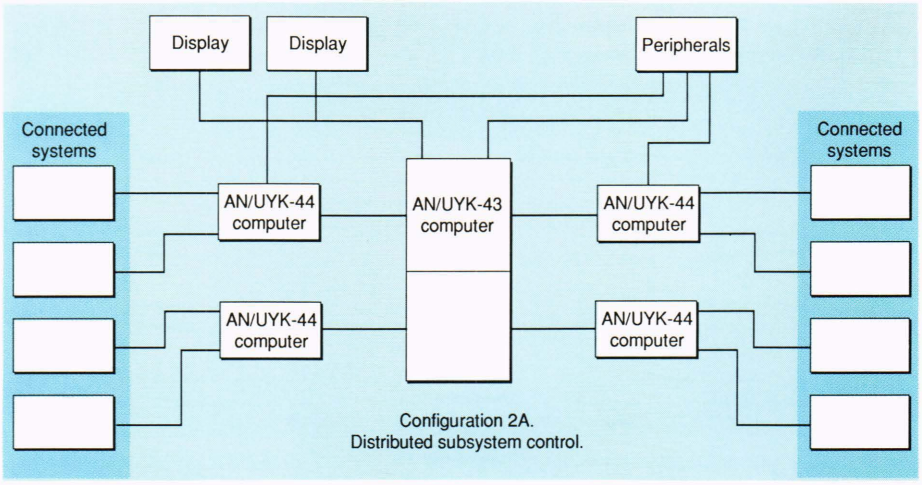


Configuration 1B.
Dual UYK-43, survivable point-to-point option.

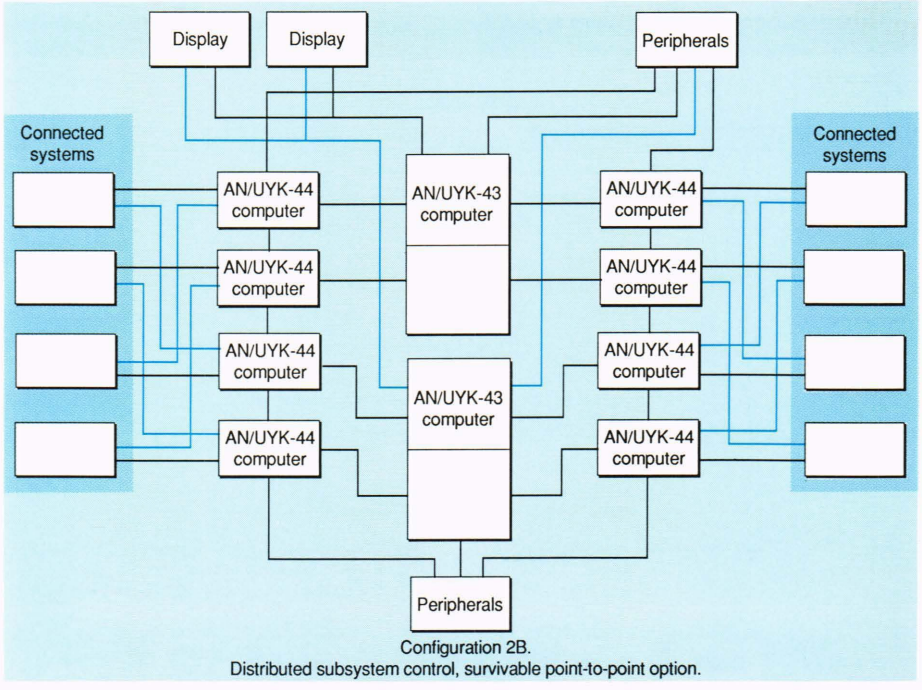


Configuration 1C.
Central control,
survivable networked option.

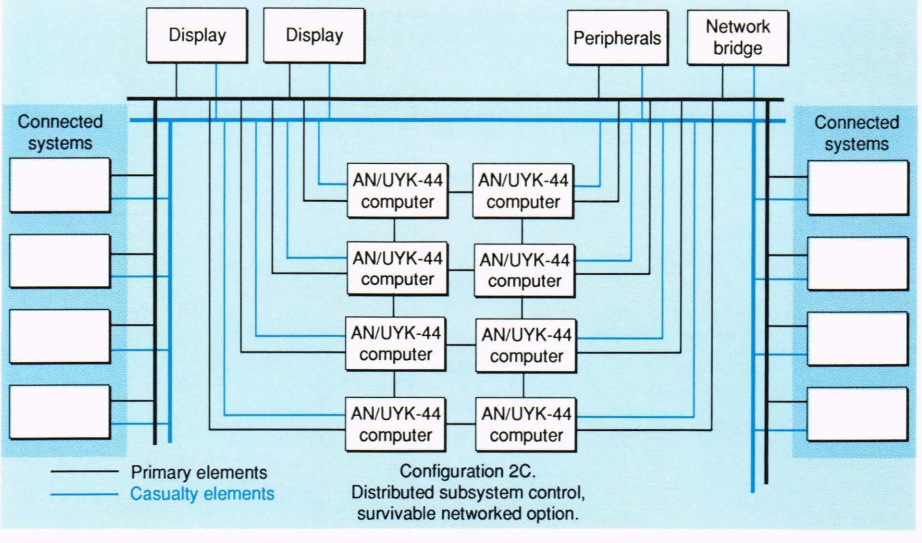
CONFIGURATION 2 OPTIONS



Configuration 2A.
Distributed subsystem control.



Configuration 2B.
Distributed subsystem control, survivable point-to-point option.



Configuration 2C.
Distributed subsystem control,
survivable networked option.

Distributed Multiple Microprocessors

The availability of high-speed local buses with compatible high-performance, single-board microcomputers has led to clustering of microcomputers, shared memory, and cards for subsystem interfacing on single-backplane buses using open-system (nonproprietary) architectures. Configuration 3A illustrates a system that uses a single cluster of 4 microcomputers and 10 interface boards on a single local bus. In a relatively small enclosure (approximately 2 ft^3), a processing capacity of 16 million instructions per second is easily achieved with industry-standard microcomputers. Recently announced processors may at least double this performance. In addition, each interface board can also include a microprocessor to permit data conversions and front-end processing for each device. Within the chassis, reconfiguration to eliminate a failed processor or interface card is possible. For major systems, a single bus may not yet adequately support data-transfer volume or the number of processors required. In addition, a single cluster is not survivable in case of local damage.

The advantages of the multiple microprocessor architecture can be realized without incurring bus overload by appropriate partitioning of the bus. Two or more clusters of microcomputers can be configured with each cluster on a separate high-speed local bus. Interconnection of processing clusters can be accomplished by any of several methods. Configuration 3B illustrates a four-cluster configuration with point-to-point connections among selected clusters.

Interconnections between clusters and connection with other combat system elements using a LAN (configuration 3C) would not only improve reconfiguration flexibility, but would significantly reduce the number of individual input/output ports (circuit boards and software) on the microcomputer clusters. The network would accommodate additional system interfaces without point-to-point cabling of each required data path. Standard network interface adapters can be incorporated into the processing cluster. To ensure that data transfers within the computer system are responsive to timing needs, selected point-to-point channels or an auxiliary network reserved for high-priority use could be added if LAN performance is not adequate to simultaneously support latency requirements of all data transfers. All data paths are redundant, including the use of multiple-network data-transfer paths. Battle damage to the computer system can be tolerated to the extent that redundant clusters and input/output cables are physically separated.

By using current technology in microcomputers, backplane buses, and networks, it is possible to implement a fully redundant computer system (every processor spared) without imposing inordinate space and weight requirements on the ship. Three or more clusters can be packaged in a single cabinet. Additional microprocessors can be incorporated into each cluster, and clusters can be added to the network to provide extensibility for future growth. Furthermore, bus-compatible and software-compatible upgrades to microcomputers of higher performance can be introduced as necessary.

RESULTS OF EMBEDDED COMPUTER SYSTEMS AT APL

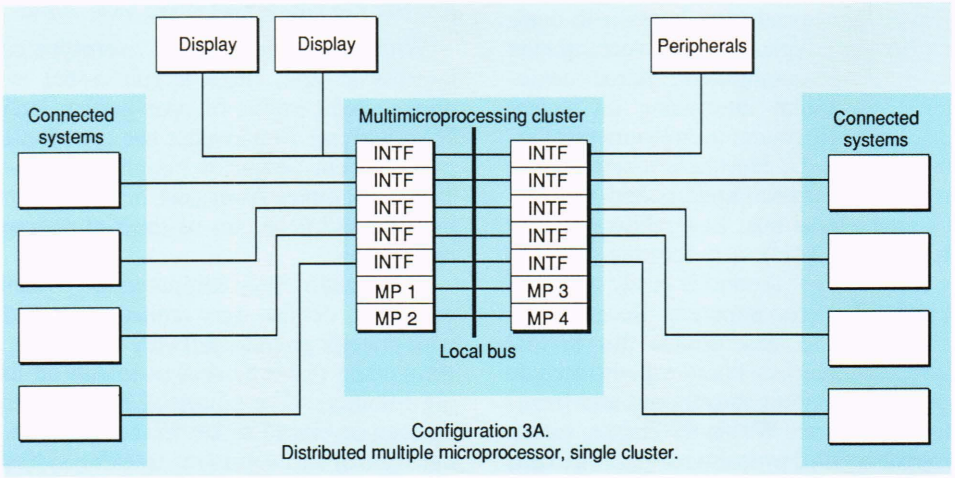
With the development of prototype computer-based systems at APL, there is substantial evidence of the above-cited benefits of open-system architectures. The Fleet Systems Department has contributed to the computer system design of major Navy systems since the 1960s; examples from past developments illustrate the benefits and problems of selected computer system architectures.

Since earlier Navy computers were relatively large and expensive, designs were limited to central architectures, with priority given to software techniques for optimizing demanding real-time designs to run on limited computing resources. The advanced multifunction array radar (AMFAR) developed at APL in the late 1960s included a radar control and automatic tracking system implemented in a single computer. It served as the advanced development radar for the Aegis AN/SPY-1 radar. Also, the Navy's AN/SYS-1 Integrated Automatic Detection and Tracking System began with an APL prototype. The AMFAR and SYS-1 computer system designs are similar to configurations 1A and 1B; they are characterized by highly efficient, event-driven, real-time executives and by extensive use of global data to minimize data transfers among modules. For these and other Navy systems, the necessity for highly optimized software designs created difficulties in software maintenance and limited the margin for upgrade.

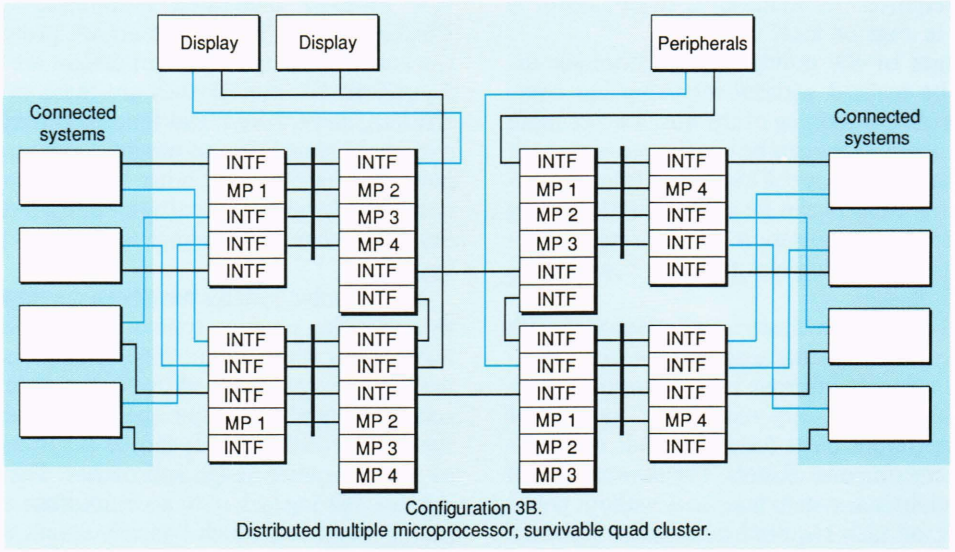
Navy combat systems have been developed and evolved by integrating separate systems, each of which, when delivered, was within about 20% of its processing capacity limit (the margin required by Navy standards). With this constraint, needs for major upgrades (requiring more than the 20% growth reserve) can be accommodated by one of two alternative design approaches. The first is to redesign the existing system to accommodate additional computing resources, which typically entails major recoding of the existing program before developing the added function. Once a program is under formal Navy maintenance, the cost of redesign, development, test, and documentation is often prohibitive. The second approach is to create a new system for each major combat system upgrade, preferably with minor impact to interfacing systems. This approach leads to configurations similar to configuration 2A and has been used successfully by APL for prototypes of the Shipboard Gridlock System[6] and Automatic Identification System for fleet use. As additional requirements develop, however, the proliferation of separate federated systems creates problems of interconnectivity, reconfiguration, and maintenance.

Recognition of the limitations of previous computer system architectures has led to application of distributed multiple-microprocessor architectures such as configurations 3A through 3C. Most real-time computer-based prototypes begun in the past three years in the Fleet Systems Department have adopted various elements of the multiple-microprocessor architectures. One of the earlier examples is the dual Motorola 68000 microprocessor-based Detection Data Converter (DDC) developed by the

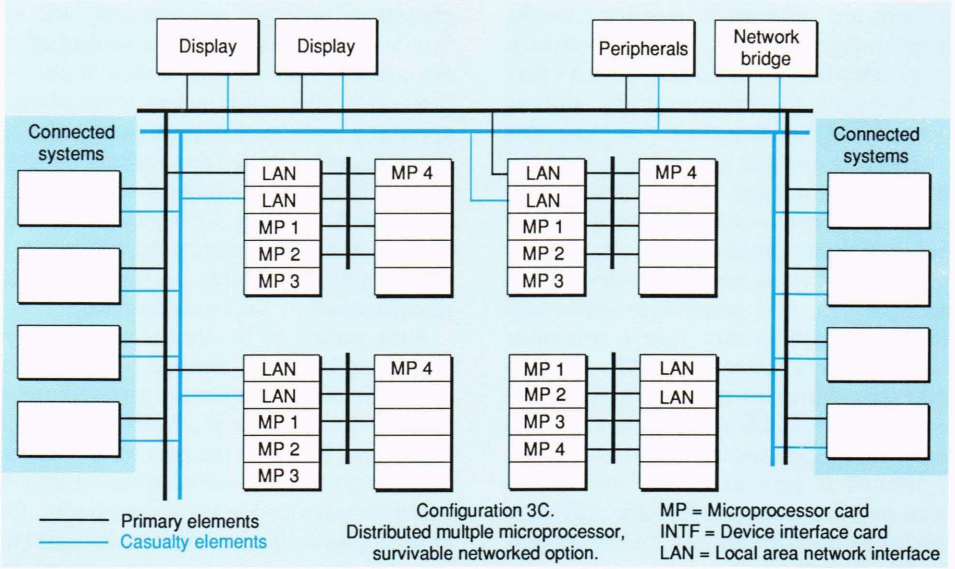
CONFIGURATION 3 OPTIONS



Configuration 3A.
Distributed multiple microprocessor, single cluster.



Configuration 3B.
Distributed multiple microprocessor, survivable quad cluster.



Configuration 3C.
Distributed multple microprocessor, survivable networked option.

MP = Microprocessor card
INTF = Device interface card
LAN = Local area network interface

Digital Systems Development Group at APL. The DDC has subsequently been incorporated into the AN/SPS-48C search radar by ITT as an upgrade to their original detection processor. Similar technology was applied on a larger scale to provide multisensor range tracking and display at the Barking Sands Missile Range. A total of 57 Motorola 68020 single-board microcomputers distributed over three radar sites and including backup redundant processors make up the computer system for the Automated Precision IFF Surveillance System. Surface and air reports are received from three sites and then are tracked, correlated, and displayed by the system as shown in Figure 2. Far more processing power than any previous Navy shipboard system in a fraction of the space and cost was clearly demonstrated.

The requirements for a Cooperative Engagement Capability had been evolving since battle group coordination concepts were formulated in 1975. At the heart of the problem was the need for a high-capacity, electronic-countermeasures-immune data link and substantial computer processing capacity on each participating ship. Analysis showed the benefits of a computer-controlled directional data-transfer system among ships of a battle group. The requirements for a prototype system, including link control, multiship/multisensor data fusion, and automatic gridlock of both track and selected detection-level data, posed a formidable computer processing task. The design that evolved is an expandable configuration of five clusters of microcomputers (Fig. 3) similar to configuration 2B that provides the required processing power and growth reserve. As shown in Figure 3, a primary bus serves processors for track management, display control, and gateways to four other buses, which serve the sensor interfaces, gridlock, and track update functions, with one bus for expansion. The system includes a message-passing method that provides a common applications program view for transfer across the VMEbus and between buses using the point-to-point gateways. The entire computer system, including peripherals and interactive maintenance panel, is housed in a 4-ft-high ruggedized enclosure. The prototype computer program and associated computer equipment are being developed by several groups in the Ship Systems Branch at APL.

In 1987, a consortium of six NATO nations agreed to undertake jointly the concept development of a revolutionary short-range anti-air defense system for their respective navies. The NATO AAW System, as now defined, includes advanced sensors of several types (radar, in-

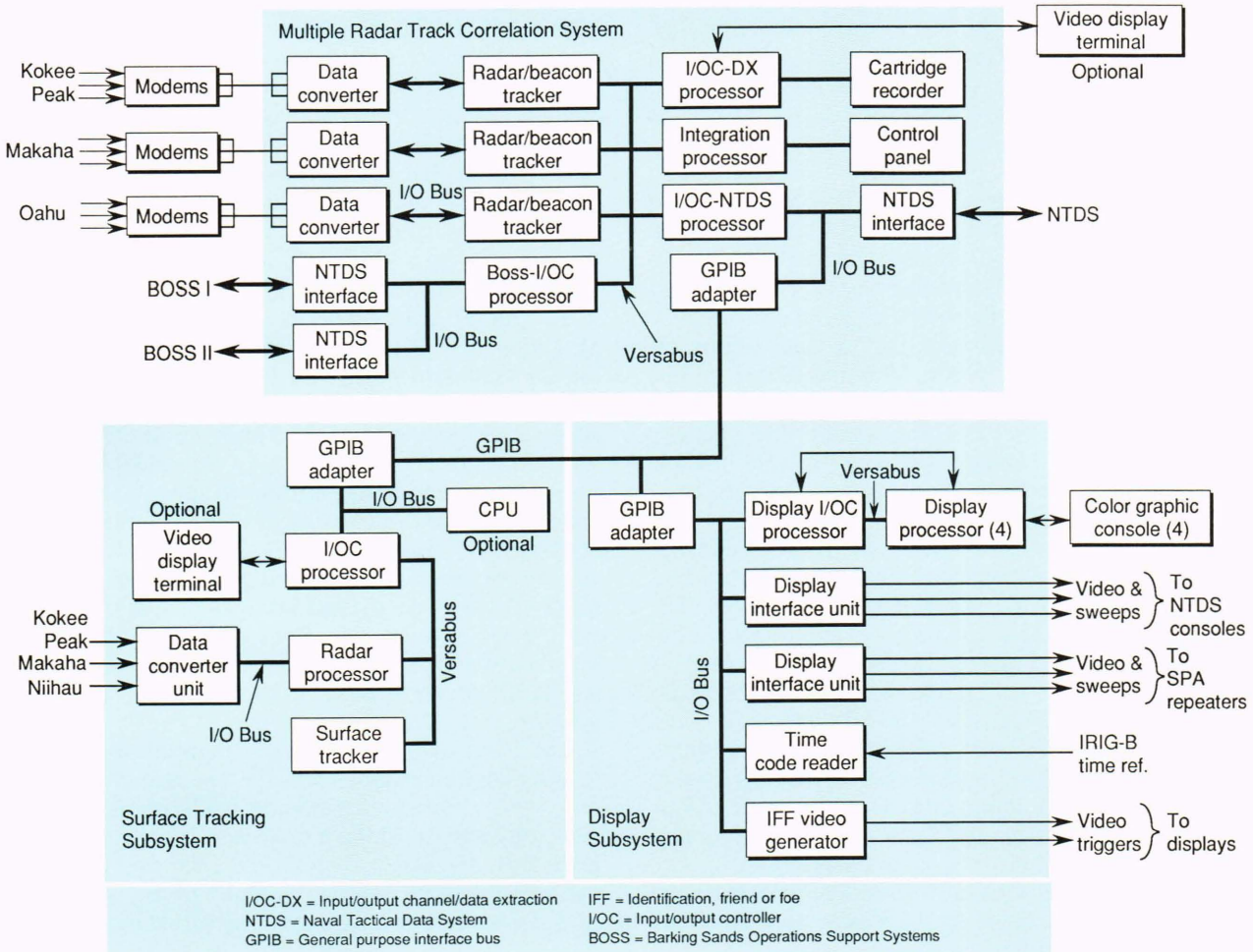


Figure 2. Automated precision IFF Surveillance System phase I/II configuration.

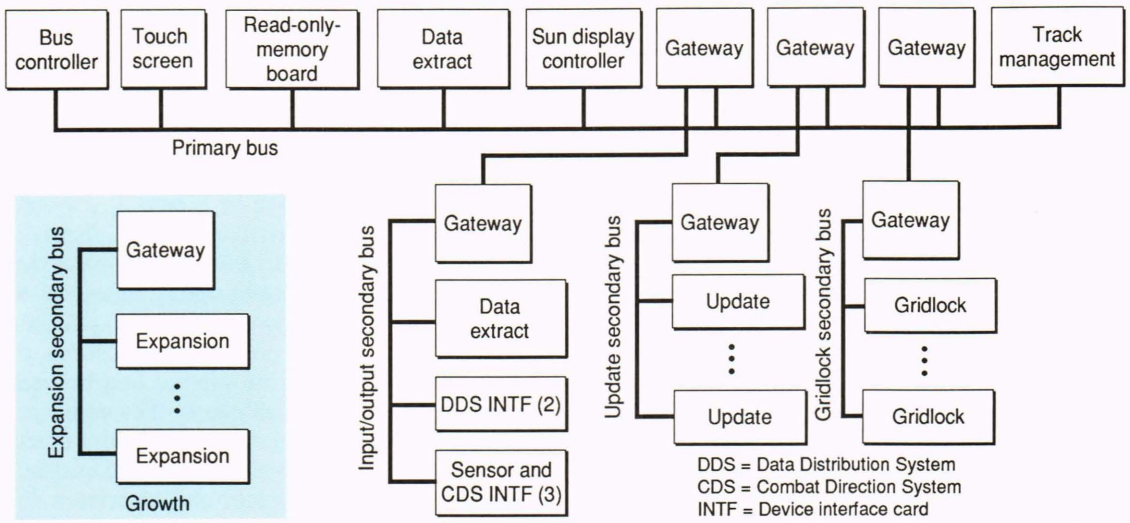


Figure 3. Configuration of prototype Cooperative Engagement Program hardware.

frared, electronic warfare support measures), a high-performance short-range missile, a control system to perform multisensor integration, hard kill/soft kill coordination, and an automatic reaction to threats based on doctrine defined in the Combat Direction System. To be effective, the computer system must perform substantially more sophisticated tracking and control processes than past systems in a fraction of the time. There is also a need to accommodate Combat Direction Systems and other interfacing systems of various navies, as well as to provide for substantial future growth and casualty reconfiguration.

One of the computer system architectures studied in depth by the international government team is the distributed multiple-microprocessor architecture similar to configuration 3C. An experimental subset was assembled at APL that consisted of three clusters of M68020 microcomputers interconnected by a triply-redundant LAN developed by the Royal Netherlands Navy. The goals of the experimental system were to examine the adequacy of the computer system architecture to satisfy the projected requirements and to investigate possible software design concepts for improving adaptability to national system variants.

At the individual processor level, the key issue was the performance of programs written in the Ada language for the currently available microprocessors. The Ready Systems' RTAda compiler was selected on the basis of run-time performance of compilers for the M68020 in late 1987. Numerous benchmarks were run to assess Ada performance with the M68020. The most widespread benchmark for Ada programs is the set of benchmarks from the Performance Issues Working Group, which is valuable in comparing the performance of various Ada compilers because it includes computational, logical, and task-control-oriented programs.

Since multisensor integration potentially poses a large processing load in the NATO AAW System, a candidate Kalman filter algorithm was implemented in both C language and Ada, providing a basis of comparison for existing sensor integration programs that have been written in the C language. The results showed that, for the compilers compared, the Ada implementation is comparable to the performance of one of the more efficient C compilers; this is in contrast to benchmarking with earlier versions of Ada compilers, which resulted in substantially less efficient code than other compilers.

Although it is expected that even higher-performance microprocessors will be used for the final NATO AAW System, the excellent performance of current processors, such as the M68020, with most recent versions of off-the-shelf Ada compilers has been comparable to processors using the more commonly used C compilers. On the basis of these findings, the use of Ada with state-of-the-art microprocessors is considered a feasible building block for the NATO AAW computer system.

To support real-time performance needs and ease of processor upgrade, it is desirable that a high-capacity, industry-standard backplane bus be used to integrate multiple microprocessors. The local backplane bus selected for NATO AAW-critical experiments and the Cooperative Engagement Program was the VMEbus, which provides a reasonably fast (theoretically 40 MB/s) bus and a wide variety of commercial boards to facilitate rapid prototype implementation. The lack of message-passing features and rigid board configuration methods, however, is viewed as a serious drawback that the Multibus II bus has overcome. Consequently, we are recommending that these features be added to the Futurebus standard, which the Navy has adopted for future systems.

The focus of the experimental development relative to data communications was the message-passing system, which provides a common applications program view for transfers within a processor, across each backplane bus, and across the LAN. Of particular interest is the decoupling of program modules by the generation of messages that can be concurrently transferred to all receiver modules registered for the message. It is anticipated that substantial independence of module design

can be achieved by this technique, thereby facilitating program development, maintenance, on-line data retention for rapid reconfiguration, and adaptability to system variants. It appears that the added processing and memory overhead required to support this concept can now be accommodated by recent microprocessor technology.

The experimental NATO AAW computer system was the first (or one of the first) Navy project(s) to integrate multiple real-time microprocessor clusters over a LAN. Although conceptually simple, there are significant technical issues associated with real-time data distribution. Once resolved, however, the benefits promise a major breakthrough in the design of combat systems and Navy software. As stated earlier, benefits include (1) extensibility by adding clusters, (2) a flexible means of reconfiguration, especially for survivability from battle damage, (3) less complex system development because of the reduced number of interfaces to each computer, (4) the potential for simultaneous access by all users because data are provided on the network, and (5) the potential for a systemwide data dictionary and associated control procedures.

SUMMARY AND CONCLUSIONS

Navy computer systems typically undergo substantial evolution to accommodate changing needs over a life cycle that may span 20 years. Advances in computer system technology now permit the use of industry-standard, open architectures that can provide for substantial extensibility to satisfy future growth needs.

Open architectures imply the use of industry-standard backplane buses versus vendor-unique bus designs, which is essential to the smooth evolution of processors, memory, and interfaces for future upgrades or to solve unforeseen performance problems during development. In most cases, careful system design, such as bus partitioning, will accommodate the performance limitations of current standards. Later upgrade to higher-performance buses (e.g., Futurebus) will better meet future needs. Compatibility with industry standards appears appropriate if alterations to withstand environmental constraints are implemented.

The need for flexible reconfiguration, particularly for local damage survivability, has led to complex methods of point-to-point interconnection in past systems. As combat system complexity increases, the only viable approach to adequate reconfigurability will be the use of LAN's to interconnect computers and ship systems. Many options remain open. Numerous industry standards exist; SAFENET I and II are Navy adaptations of token passing-ring standards.

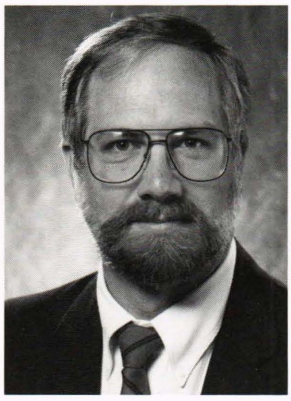
Multiple high-performance processors coupled with effective underlying data communications techniques—backplane buses and LAN's—offer the framework for a breakthrough in the design of Navy software that could greatly simplify the development and life-cycle evolution of large systems. By sufficiently decoupling functions, removing dependency on shared data, and providing tight task synchronization, a level of modularity can be achieved that has heretofore been unattainable. The provisions of Ada packages and object-oriented design techniques are compatible with these goals.

REFERENCES

1 *AN/UYK-43 Technical Description*, UNISYS (1987).
2 *AN/UYK-44 Technical Description*, UNISYS (1987).
3 *MC68020 32-Bit Microprocessor User's Manual*, Motorola, Prentice-Hall, Inc. (1985).
4 James, J., "Performance and Programmability Considerations for VMEbus Multiprocessors," in *Proc. BUSCON East*, DY-4 Systems Inc., pp. 225–239 (1987).
5 Marien, K., "Emerging Standards, Hardware and Software Light the Way to FDDI," *Comp. Des.*, 51–56 (1989).
6 Miller, J. T., and David, E. W. G., "Battle Group Gridlock Demonstration," *Johns Hopkins APL Tech. Dig.* **2**, 314–320 (1981).

ACKNOWLEDGMENT—The author acknowledges the technical contributions of Eric Conn, Robyn LeGrys, Larry Norcutt, and John Zouck.

THE AUTHOR



JAMES G. PALMER, supervisor of the Computer Systems Development Group in the Fleet Systems Department, received a B.S.E.E. degree in 1966 from Drexel University and an M.S. degree in numerical science in 1970 from The Johns Hopkins University. Since 1966 he has been employed at APL, working in the areas of radar control, sensor integration, real-time systems design, and software engineering. His experience includes development of radar control software for a prototype of the Aegis AN/SPY-1 radar, development of an experimental Aegis sensor integration program, and supervision of prototype system development for the Battle Group Anti-Air Warfare Coordination and NATO AAW programs. In 1975, Mr. Palmer participated in the Department of Defense Weapon Systems Management Study. He has been an instructor at The Johns Hopkins University G. W. C. Whiting School of Engineering.