cockpits. For instance, if one were evaluating pilot response to aircraft roll rates or analyzing training effectiveness, full motion-based simulation is often necessitated. But, control and display research may not necessarily be so task intensive. New ideas and seeds of new ideas can be screened at a more basic level in order to advance these technologies. When this basic research is conducted in a mission context with an out-the-window-view, the simulation is indeed enhanced; however, there is some doubt whether it makes sense to tie up high fidelity simulators at this stage of development. Not only are costs tremendous, but the resource flexibility needed at this level does not exist when using large simulation facilities.

The Flight Dynamics Laboratory has taken advantage of the micro-boom to design and develop a fixed-based, dynamic cockpit which provides the capability and flexibility to conduct control and display research over a broad range of experimentation. This cockpit is known as MAGIC, which stands for Microcomputer Applications of Graphics and Interactive Communications, and is shown in Fig 1.

Figure 1. MAGIC Cockpit

MAGIC Configuration

The MAGIC cockpit is designed around four CompuPro 8086/8087 microcomputers using the IEEE-696 (S-100) bus. The workload for each microcomputer is distributed according to the specific function it performs. The value of this system configuration is directly related to flexibility and expandability; it can easily be reconfigured or expanded to satisfy requirements. In addition because of its modular design, the tremendous costs associated with total system replacement are avoided; instead the new requirements can be fulfilled by upgrading the system. For example, if more memory is needed, a board with additional memory can be purchased and plugged into one of the slots in the micro chassis. However, if additional speed and number crunching power is required, another microcomputer can be added to this system that works in parallel with others.

Before functionally breaking down the tasks to be performed by each of the microcomputers, the network had to be selected. When designing our microcomputer network, the two factors which received our greatest attention were each of the micro's workload and the I/O communications between the micros. Considering I/O, the question of which communication link would be used between the micros had to be answered. Was high speed (parallel) communication necessary or would the lower rate of data transfer (serial RS-232C) be sufficient? Two factors which affected the decision were quantity of data sent and the update rate needed for this data. RS-232C communication links at 19.2K baud were selected for interprocessor communications, because the input was from a slow source, the human. However, the Electronic Attitude Director Indicator (EADI) used by the pilot for flight direction feedback requires instantaneous updates. Therefore, the stick and throttle information to the aeromodel had to be along a high-speed parallel link.

For an overview of the cockpit and system configuration refer to Fig 2. The major hardware modules comprising the MAGIC simulator are:

1. Four Compupro 8086/8087 microcomputers.
2. Votan V5500A Speech Recognizer.
3. MicroAngelo Scion 5020 Color Graphics System.
4. Gaertner Graphics System.
5. Four Pioneer PR-7820 Model 3 random access Video Players.
6. Five color cathode ray tubes (CRT's).
7. Twelve Microswitch Programmable Display Pushbuttons (PDP's).
8. Bowmar programmable Multifunction Control (MFC).
9. Elographics touch sensitive overlays.

The great majority of functional software is written in Pascal, with the remainder being written in FORTRAN. The hardware-specific I/O drivers are written in Assembly language. The system operates under CP/M-86.

After completion of the system analysis, the tasks selected for each of the four microcomputers are as follows:

Microcomputer 1

Micro #1 is the overall system executive, responsible for coordinating the start and stop of the simulation. As Fig. 2 indicates, this micro handles the interface and feedback to the pilot. The Test Operator's Console (not shown in Fig. 2) is also controlled through Micro #1. Furthermore, all data generated for later analysis, such as the sequence of switch hits and the task duration times are collected by Micro #1.

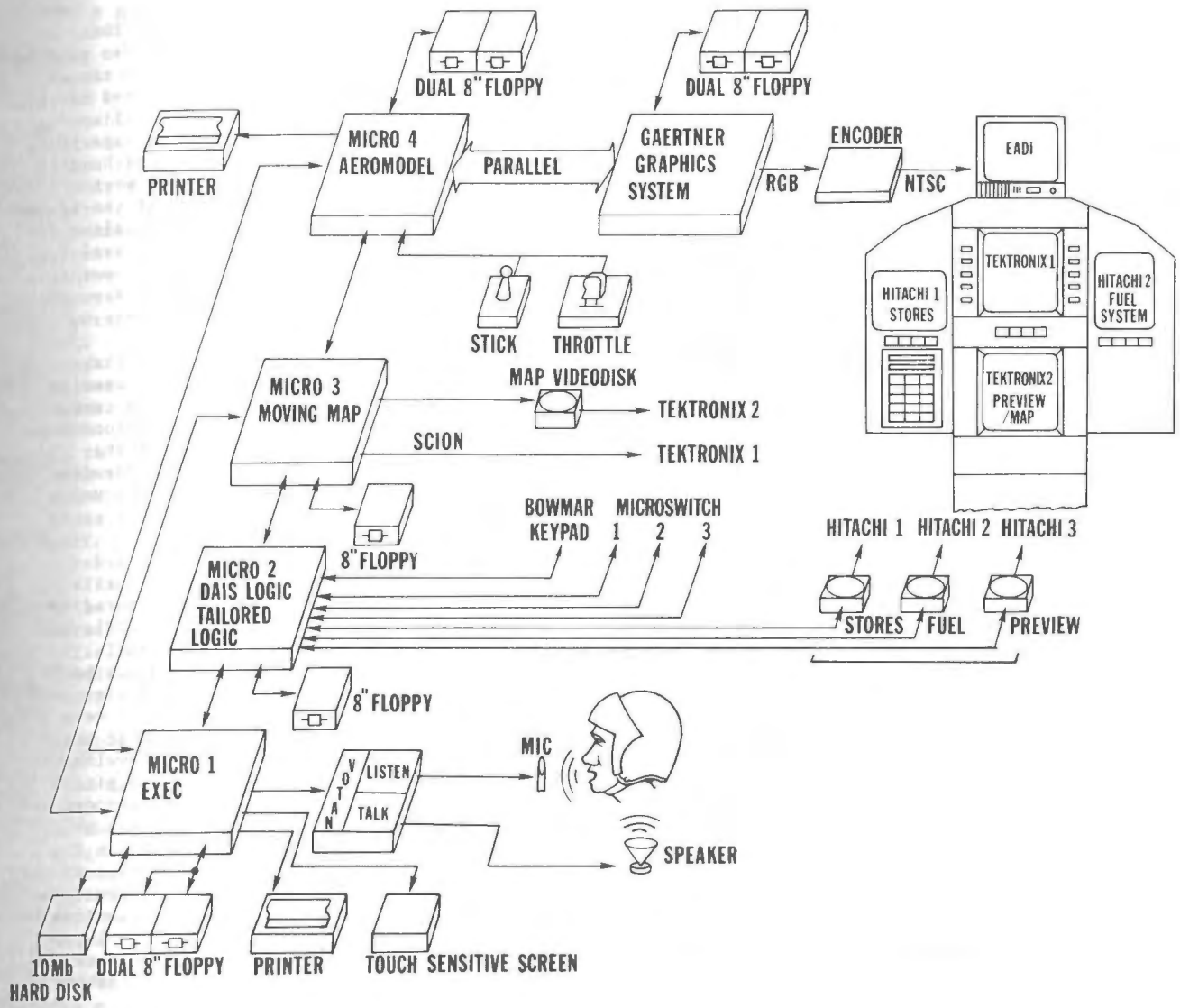


Figure # 2 MAGIC Configuration

Microcomputer 2

Micro #2 contains the logic trees that tailor the programmable microswitches and Bowmar keypad output, based on the pilot's input. This micro also controls three of the four videodiscs showing: systems status, stores status, and a three-dimensional preview of a target.

Microcomputer 3

Micro #3 controls the graphical moving map display that is controlled by the pilot's latitude and longitude. This display is used by the pilot as a look-down view of his track, the surrounding terrain, and his true aircraft position relative to the track and terrain. Micro #3 also controls one videodisc which shows a map of any waypoint he selects, followed by a photo of either a 180 degree or 270 degree view of that waypoint. Last of all, this micro controls cursor movements to identify a pop-up target using a touch sensitive screen overlay, a manual switch, or voice control.

Microcomputer 4

Micro #4 contains the aeromodel and flight director, as well as the routines for data gathering and post-processing of the primary flight data. The EADI display previously mentioned is driven by Micro #4, and is generated by the Gaertner Graphics System.

Example Studies

The MAGIC cockpit can be used to conduct studies of various levels of sophistication. Two examples will be provided, one illustrating a relatively high level of sophistication and the other a less sophisticated effort.

Example 1: Applied Tailored Logic and Speech (ATLAS)

One critical consideration for designers of current and future aircraft cockpits is the placement of the myriad of controls and displays necessary to operate a modern weapon system. Simply finding panel space to accommodate new equipment is becoming a major problem. Two of the more widely accepted methods of addressing this situation are through the use of multifunction controls (MFC's) and voice recognition systems. The use of voice control reduces the need for manual controls to be located within a pilot's immediate visual and physical reach envelopes; only the activation switch has to be within fast access range. An MFC addresses the problem of cockpit space by allowing several systems to be operated from the same control device, simply by changing the legends on the various switches to those necessary for a particular system. This reduces panel space by controlling most systems from the single MFC panel. Fig. 3 shows the MFC installed on the lower left panel of the MAGIC cockpit.

The study discussed in this paper was designed to compare voice recognition versus the MFC for controlling aircraft subsystems. In addition two types of system control logic were used -- Branching Logic and Tailored Logic. The Branching Logic menu tree structure starts with the highest level function (e.g., communication) then proceeds to the next level (radio types) then goes to the submodes of the radio (frequency change). Tailored Logic, on the other hand, is not organized along system functional lines, but rather according to aircraft flight phases. For example, in the cruise phase the most likely used subfunctions, be they navigation, communication or avionics, are all available at the highest level in the tree. This eliminates the time consuming step of proceeding through several levels of system logic to access a commonly used function. The ATLAS study was performed to compare the performance of this Tailored Logic to the standard Branching Logic, using both the MFC and voice controlled systems.

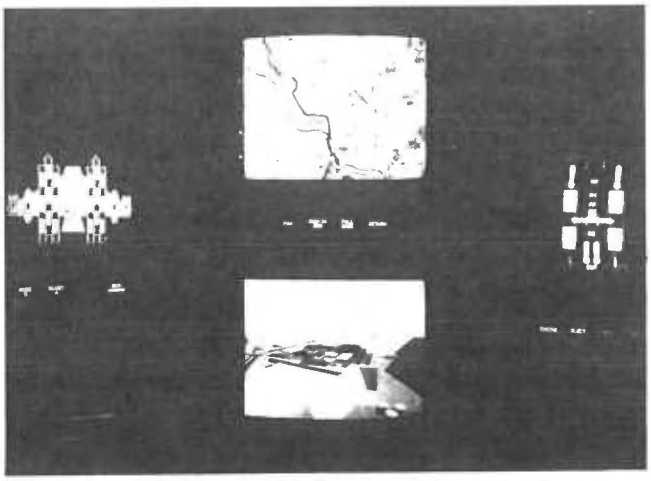


Figure 3. MAGIC cockpit front panel with a multifunction control on the lower left hand side.

How the Study was Conducted

In order to increase their workload, the subjects were required to "fly" a video game in addition to using voice or the MFC to control aircraft subsystems. The subjects used in this study were 18 Air Force personnel, all having had considerable prior video gaming experience. Potential subjects were give one-half hour to practice on the video game and then were required to play a test game against the system in which they had to keep five ships alive for at least an average of 30 second survival time per ship. If the subjects could not meet this minimum criterion, they were dropped from the study.

To illustrate the flexibility of our micronetwork, let's look at the equipment used in this experiment. The ATLAS study was conducted using two of the four (#2 and #4) microcomputers available. All CRT's displaying the computer generated graphics or the videodisc pictures were used, in addition to the voice recognition system, to accomplish this study.

The two types of control logic discussed earlier were displayed on the programmable display pushbuttons (PDP's), manufactured by Microswitch and the Bowmar MFC. The PDP's are matrix-addressable, each with 560 pixel, light-emitting diodes (LED's) which can be programmed for both alphanumeric and pictorial legends. Only alphanumeric characters were used. The color of the LED elements in the switches is green, with a dominant wavelength of 568 nanometers; the nonilluminated pixels appear black. Referring to Fig. 3, you can see the three sets of four PDP's underneath three of the CRT's. The Bowmar MFC located on the bottom left panel of the cockpit is also an LED programmable device. Data entry for new radio frequencies or the weapons configuration can be made dynamically from the Bowmar. Its layout is a 3 X 5 addressable area, with the scratch pad area on the top used for current tasking feedback.

Discussion

The voice and MFC modes, both using Tailored Logic involved fewer inputs than the Branching Logic to accomplish a given control operation, and this difference manifested itself as a reduction in overall control operation time. The two control modes under Tailored Logic also required fewer glances at the face of the MFC to either ascertain switch positions or confirm correct control operation, resulting in greater ability to focus subject attention on the video game loading task. This is evidenced by the significant reduction in subject ships destroyed during control operations. Because the subject's hands could remain constantly on system controls, task initiation time with the speech system was significantly shorter than with the manually operated systems. Error rate was predictably higher in the branching mode, primarily because the more complex series of switch selections allowed the subject both more opportunity to become lost in the control logic

and a greater probability of making simple control selection errors.

Results of this study indicate that multifunction controls, with properly designed tailored switching logic can be as effective as voice for the control of cockpit systems. In this case, performance for both systems was essentially equal; however, it should be pointed out that voice control still has some distinct advantages over MFC's for some tasks not examined in this study. For example, voice allows the pilot to scan a 360° view outside the cockpit and still maintain control of aircraft systems; this is not possible with the MFC. In the current study, only one parameter (task initiation time) differed significantly between the voice-operated and tailored-manual modes.

It is important however, not to ignore the very real improvement produced in the operation of the MFC by the implementation of the Tailored Logic. The new logic significantly improved performance over branching control logic in three of the six metrics used in the study (kills within tasks, task time, and error rate) while performing equally well in the other three. Clearly the Tailored Logic is much more efficient than the Branching for those tasks immediately available on the MFC, as well as being more acceptable to the user.

Example 2: Programmable Switch Study

This second example used only a portion of the MAGIC cockpit but still provided valuable data on the use of new cockpit technology -- the programmable display pushbuttons (PDP's) discussed in the last example. Although the pictorial capability of the switches was not utilized in the ATLAS study, it is this aspect which is of special interest in this study since it can provide an additional means of information transfer between the operator and the machine. However, as is the case with any newly available technology, research is needed to determine how to effectively employ the pictorial display aspects of the switches. A review of the research conducted indicated that there may be three general classes of pictorial symbols: those which are intuitive at first glance by the untrained subject; those which are not intuitive at first glance but become so after training; and those which are never intuitive regardless of the extent of training. The purpose of the study was to test the intuitiveness of the following twelve pictorial symbols: tank, dam, tunnel, water, bridge, train, surface-to-air missile (SAM), anti-aircraft artillery (AAA), petroleum-oil-lubricants (POL), convoy, armored personnel carrier (APC), and troops. The intuitiveness was measured by comparing the glance comprehension of the twelve symbols.

How the Study was Conducted.

Subjects were paid student volunteers who responded to solicitation in the campus newspaper. The total number of subjects employed was 24, with half being male and the other half female.

Only one microcomputer (#2) was needed thus allowing software development for the follow-on study to continue in parallel with little interruption on the remaining three micros. None of the CRT's in the cockpit was used, and only one of the twelve programmable switches was needed. Furthermore, the voice control capability and Bowmar MFC were not included.

In the past when a minicomputer was used, time had to be scheduled for its use. Either the programmers would be sitting idle, with no system available during an experiment, or the experiment would be extended so the programmers could make real-time fixes or continue with another effort. Now the programmer and experimenter are both satisfied with this flexible system.

Before seeing the experimental symbols for the first time, each subject was given a total of four familiarzation examples of a symbol which was displayed on the switch in the same manner and for the same duration as during the experimental trials. The training symbol, a house, was not one of the twelve symbols included in the experimental set. After each subject was comfortably seated in the cockpit, a small square was presented in the center of the switch to serve as an attention focus point. This alerting stimulus lasted for 500 milliseconds (msec). Then the switch blanked out for 500 msec and the target stimulus appeared. The duration of the target stimulus was 43 msec and there was a 7 msec delay between the end of the target stimulus and the onset of a masking stimulus which lasted for 300 msec. The instrument panel of the cockpit was masked with flat-black foamcore, leaving a single programmable switch in the center of the panel visible to the subject. The approximate distance from the switch surface to the subject's viewpoint was 29 inches.

Discussion

The criterion set for the intuitiveness of a symbol was a 90% recognition rate. The 90% recognition criterion took into account the very short exposure duration of 50 msec. In an actual aircraft environment, the viewing time would more likely be a few seconds, with recognition accuracy correspondingly increasing. If a symbol achieved this 90% recognition rate on the first exposure, it was placed into the intuitive without training category. If it did not achieve it on the first exposure but did after the subject had been thoroughly trained as to its meaning, it was placed in the intuitive after training category. And if it never achieved the 90% recognition rate, it was placed in the non-intuitive after training category.

In the intuitive without training category, only one symbol qualified (troops with a 96% recognition rate). In the intuitive with training category, all of the symbols but the tunnel qualified, with recognition rates ranging from 92% to 100%. The tunnel had only a recognition rate of only 62% even after training. This means the tunnel fell into the

last category of non-intuitive even after training and must be redesigned.

Conclusion

Microcomputers have dramatically affected the cockpit designers' research. No longer are they bound by the cost constraints associated with mainframe facilities. The flexibility offered by the relatively inexpensive, distributed microcomputer system provides researchers with a means of conducting cockpit studies at varying levels of fidelity. As shown in the switch study, this micro-network flexibility allowed software development to continue during an experiment since the full system was not being utilized, unlike a mini or main-frame computer. The MAGIC facility discussed in this paper illustrates a low fidelity system. As hardware costs decrease and the power of micros continues to increase with very high speed integrated circuit (VHSIC) technology, before too long they may be as powerful as a CRAY computer, thus limiting researchers only by their imagination.

References

(1) Canon, J. Toward a Totally Integrated Aircraft. Airforce Magazine, December 1983, 34-41.

(2) Amico, V. and Clymer, A. B. Simulator Technology - Forty Years of Progress. Simulation Series, 14(1) La Jolla, Calif., 1984.

(3) Gravely, M. L. and Hitchcock, L. The Use of Dynamic Mockups in the Design of Advanced Systems. Proceedings of the Human Factors Society, 1980.

(4) Lizza, G. D., Howard, B. and Islam, C. MAGIC - Riding the Crest of Technology or Do You Believe in MAGIC? Proceedings of the Human Factors Society, 1983.

CERTIFICATION OF A HOLOGRAPHIC HEAD-UP DISPLAY SYSTEM FOR LOW VISIBILITY LANDINGS

John P. Desmond
Vice President, Engineering

Douglas W. Ford
Principal Control Systems Engineer

Flight Dynamics, Inc.
Hillsboro, Oregon

Abstract

The purpose of this paper is to summarize the approach taken to achieve certification for operations in CAT III weather minimums (down to 700 ft runway visual range) through guidance information presented on a single Head-Up Display. The paper discusses the original strategy designed to meet FAA requirements, the effect of these requirements on the system design, and additional requirements imposed by the man-in-the-loop and the target aircraft. System architecture and aircraft sensor requirements are outlined. The simulation and flight test program are described, and some test results are provided.

Summary

Since this was the first pilot-in-the-loop CAT III system, no guidelines or specific requirements for certification existed at the beginning of this program. Guidelines did, however, exist for the approval of CAT III landing weather minimum, AC 120-28C (1) and for the approval of automatic landing systems, AC 20-57A (2). AC 120-28C addressed the possibility of certifying CAT IIIa operations with "the pilot-in-the-loop active-control if Proof-of-Concept testing demonstrates that these systems provide an equivalent level of safety". Limited information existed on the definition or scope of Proof-of-Concept testing. From AC 120-28C:

> "Proof of Concept Testing. Proof of concept testing is defined as a generic demonstration in a full operational environment of facilities, weather, crew compliment, aircraft systems, environmental systems, and any other relevant parameters necessary to show concept validity in terms of performance, system reliability, repeatability, and typical pilot response to failures. Proof of concept may be established by a combination of anaysis, simulation and/or flight demonstrations in an operational environment."

Initially we expected that the entire certification program would demonstrate Proof-of-Concept and no special test program would be required. After the experience gained in acquiring CAT I and CAT II STCs we realized the importance of a Proof-of-Concept program to resolve the issue of the acceptability of a single HUD for CAT III operations, and to resolve any crew system interface problems in the simulated environment. This program proved most useful in resolving these issues prior to beginning the expensive certification simulation and flight test verification programs. Additionally the Proof-of-Concept tests demonstrated the systems ability to perform to CAT III requirements without the installation of an autothrottle.

We also received upon application for STC a list of highlighted Federal Aviation Regulations (FARs), Advisory Circulars and Approval Criteria from the FAA which assisted in further defining the problem.

This program was the first attempt to meet CAT III performance requirements without the direct involvement of the airframe manufacturer. This presented additional problems including: establishing compliance to structural requirements for the installation of the overhead and combiner units; 2) the availability of a complete and verified simulation of the aircraft; 3) establishing the aircraft installation operation, and maintenance procedures; and 4) the performance of aircraft installation, operation and maintenance.

In summary the problem encompassed:

1. Development of a pilot-in-the-loop control system and display that would meet the performance criteria for automatic landing systems while maintaining an acceptable level of pilot workload.

2. Design of a HUD system that could execute this control program and provide it the necessary sensor information while meeting the safety requirements imposed on CAT III operations. These requirements drove system architecture, software and hardware design criteria.

3. Development of the system/crew interface and crew procedures compatible with CAT III weather minima and the HUD system.

4. Design of an aircraft installation drawing package which would ensure consistency of installation on multiple aircraft. This included development of aircraft wire separation guidelines for aircraft not originally equipped for CAT III operations. The target aircraft was a Boeing 727-100.

5. Preparation of safety and failure mode and effect analyses to ensure the system would meet CAT III safety requirements.

6. Preparation of simulation and flight test plans for system development, evaluation and verification.

7. Selection of a flight test crew for simulator and aircraft programs.

8. Development and installation of the flight data acquisition and touchdown verification equipment.

9. Aircraft acquisition, installation of equipment and operation of the flight test aircraft. Two Boeing 727's were involved.

10. Preparation of final reports and analysis and demonstration of aircraft and simulation performance.

Early, it was decided to develop a system which would meet the requirements with a single HUD and to put this HUD on the Captain's side. A dual HUD system only made sense with both pilots head-up during the approach and landing, requiring that all the panel information, including engine, navigation, and warning be integrated into the HUD symbology. Providing all the panel information head-up presented the difficulty of establishing display formats for a vast amount of information while maintaining approach and landing symbology conformal with the outside world. Also, a head-down right seat pilot has access to aircraft system information and is able to monitor conventional instruments providing dissimilar redundancy in evaluating approach progress.

2. HUD System

System Configuration The pilot looks through the holographic combiner, Figure 3, to acquire guidance and situation information focused at infinity. The overhead optical assembly in conjunction with the combiner present symbolic images to the pilot projected from a cathode ray tube at the rear of the overhead assembly. A drive electronics unit provides CRT drive functions. Mode selection and data entry are accomplished through a HUD control panel. The HUD Computer performs symbol generation, executes the guidance algorithms, provides the monitoring functions, establishes the interface with the aircraft sensors and evaluates input data. A system functional diagram is presented in Figure 4.

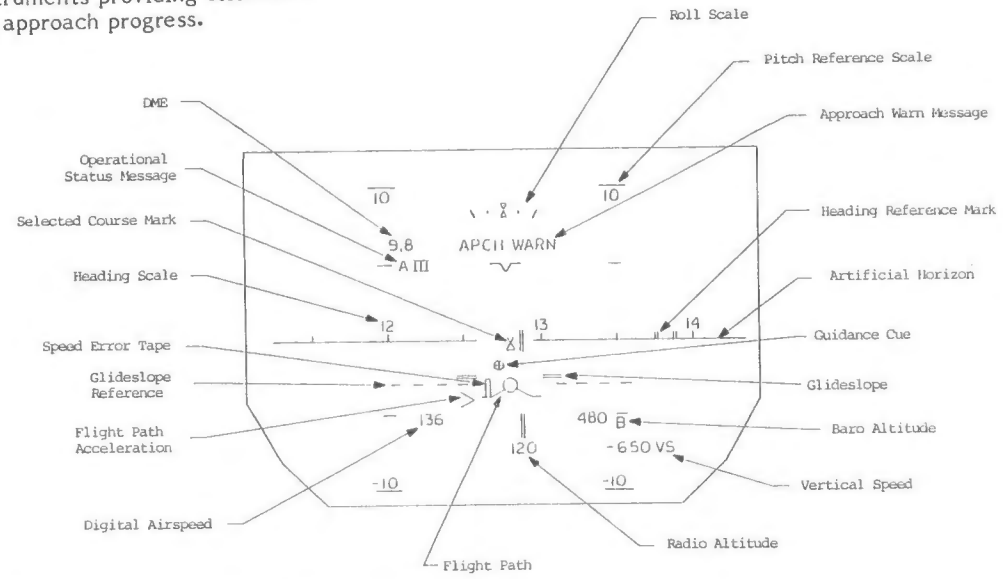


FIGURE 1
HUD SYMBOLOGY

1. Development Of The Pilot-In-The-Loop Control System

Fundamental to the acceptance and success of this system, was the ability of its display to provide the pilot with information sufficient to accomplish hand flown landings meeting the touchdown criteria for CAT III operation. This capability was provided the pilot via presentation of an advanced flight director display. The director provides the pilot compensatory aileron and elevator commands for flight from localizer capture to touchdown. The pilot's primary task on approach is nulling the flight director, and secondarily nulling the airspeed error. The pilot's workload, his opinion of the system, and his performance are largely determined by the flight director control law. Extensive use was made of prior investigations [3] of pilot behavior and the theory of manual control [4] in the design of the FDI flight director. The insight provided in the crossover model [5] and related describing function analysis of pilot-vehicle-display characteristics was proven invaluable in modeling and analysis of the HUD system. "K/s like" dynamics in the region of crossover was designed into the director system with excellent results as predicted by the theory. The Director system has proven to be easy to fly and to meet accuracy requirements for CAT III operation. The symbol set provided the pilot is illustrated in Figure 1, and shown in the photograph taken in flight in Figure 2.

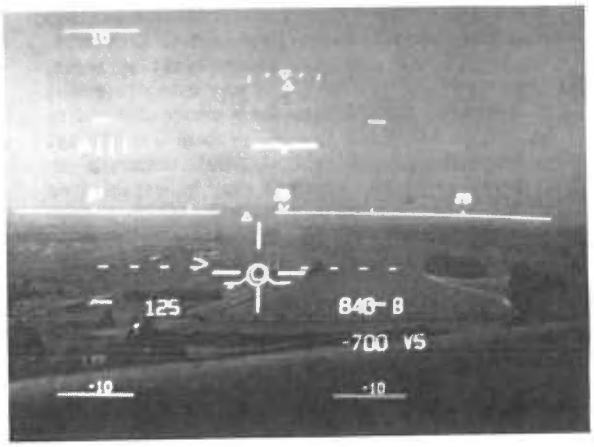


FIGURE 2
APPROACH SYMBOLOGY

FIGURE 3
AIRCRAFT INSTALLATION,
COMBINER AND OVERHEAD UNIT

The HUD Guidance System (HGS) consists of the HHUD units and the aircraft sensors and systems shown in Figure 5. An inertial reference unit (IRU) provides the precise attitude and heading information necessary to ensure symbology and real world conformality, and in addition, ground speed, inertial vertical speed, and track angle necessary to display accurate flight path information. The IRU installed in the test aircraft is a Honeywell laser gyro unit. Vertical and directional gyro inputs are used as comparator inputs along with IRU data to catch any unflagged IRU errors. Inputs from two central air data computers provide comparison monitoring for airspeed and barometric altitude. Dual inputs are also provided from radio altitude and the localizer and glideslope signals of the Instrument Landing System (ILS).

HUD Computer Four independent microprocessors execute HUD System functions. All four are programmed in high level language to facilitate documentation and provide visibility of software functions. This is especially important in reaching the fail-safe criticality levels dictated by the system application. Software has been developed in accordance with the requirements of RTCA document DO-178. Two independent microprocessors perform data acquisition and provide channel separation of input data. A 16 bit micro processor, the Control Law Processor, performs the data comparisons from the two input channels, executes the guidance algorithms and drives the symbol generator. A second 16 bit micro, the System Monitor, also accesses dual input data and performs the monitor functions. This monitor micro can cause the display of warning messages, deletion of suspect data or blanking of the entire display.

Holographic Combiner The pilot views system symbology and the outside or real world through the holographic combiner. Two pieces of optical glass form the plano-plano combiner with the holographic element sandwiched between. The combiner is the primary collimating element of the optical system.

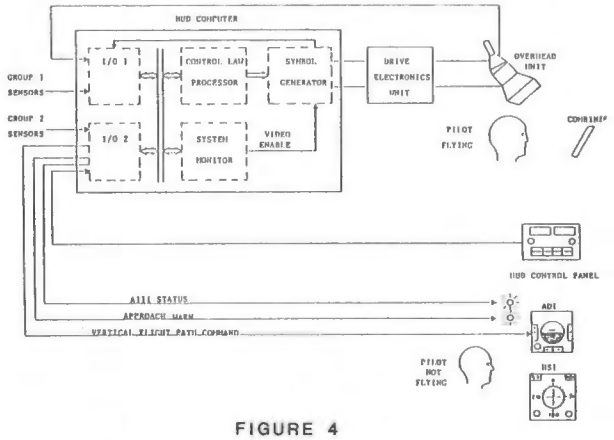


FIGURE 4
HUD GUIDANCE SYSTEM CAT IIIA

The HHUD optical system which includes the combiner and overhead relay lens provides a 30° horizontal by 24° vertical overlapping field-of-view for symbology display. This wide field-of-view is necessary to accommodate symbology shifts from boresight due to flights in crosswind conditions.

System Monitor To ensure that no undetected failure will cause significant deviations from the approach path or touchdown footprint an independent system monitor has been implemented in the HUD computer. The monitor provides two functions. First, it verifies placement of symbology on the display to detect any misrepresentation of critical situation or guidance information. The purpose here is to identify and blank any information which could cause incorrect control movements by the pilot. The System Monitor verifies operation of the Control Law Processor, Display Generator, Drive Electronics Unit

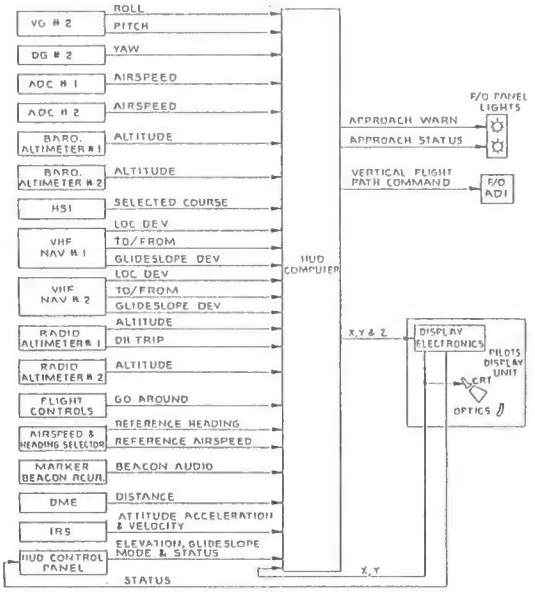


FIGURE 5
HUD SYSTEM

and Overhead Unit. To eliminate the possibility of a similar latent software error in the System Monitor and the Control Law Processor, a dissimilar set of algorithms are implemented in System Monitor software.

As a second function, the System Monitor assesses the approach progress and annunciates a caution to the crew if the approach exceeds limits which could cause a landing outside the desired touchdown footprint.

3. System/Crew Interface and Crew Procedures

The ability of the pilot not flying (PNF) to monitor the progress of the approach through the initiation of flare while remaining head down was a primary consideration during system design. The following information provided to the PF on the HUD is also provided on the PNF instrument panel to ensure awareness of system status and approach progress.

1. An annunciator indicating system readiness to execute a CAT III approach (AIII status).
2. An Approach Warn indicator which annunciates a system malfunction or an approach flown outside required limits.
3. An expanded localizer display at the bottom of the ADI with indicated limits equal to localizer limits on the HUD.
4. The vertical guidance command which the PF is following on the HUD is repeated head down for the PNF to enable him to evaluate pilot tracking performance below the altitude where glideslope information becomes unusable.

Two other duties of the PNF are 1) assisting in performing the approach checklist and 2) executing a go-around in case of a HUD system failure. Prior to approach initiation, the approach course, airspeed, field elevation and glideslope angle are entered through the Captain's panel instruments and HUD Control Panel. By initiating a system test, these values are displayed on the HUD, and then read by the PF and cross-checked to the approach plate by the PNF. Below 500' the PNF rides passively on the controls; one hand on the wheel and the second behind the throttles. A "go-around your controls" call by the PNF gives control of the aircraft to the PNF. He will take control without direction only on loss of response by the PF in a response exchange initiated prior to decision height.

4. Aircraft Installation Drawing Package

To ensure consistency and repeatability of system installation a detailed drawing package was developed prior to intial aircraft installation and revised during installation on the two flight test aircraft. Achieving system separation on the aircraft in order to prevent loss of critical information to both the Captain and First Officer in the event of a wire bundle overheat or serious avionics malfunction required rework of existing aircraft wiring in the form of physical wire separation or sleeving where separation was not possible. A document describing the wire separation objectives and methods was prepared and approved by the FAA prior to system installation in the No. 2 aircraft.

The drawing package also contains detailed installation instructions, an operational test procedure to ensure a proper interconnect to all avionics systems and a compatibility test to ensure non-interference with other aircraft systems.

5. Safety and Failure Analyses

The ability of the system to meet the safety requirements for operation in CAT III conditions was established and verified by conducting three separate analyses each resulting in extensive supporting documentation.

The System Safety Analysis considered the ability of the system to ensure that the probability of undetected hazardous events were on the order of one part in 10^{-8}. The following hazardous events were included in the analysis: long landing, short landing, landing laterally off the runway, hard landing, wing tip strike. and nose wheel strike. The conditions which could lead to these events were then examined. Once the causes of each hazardous event were identified, fault trees were constructed in accordance with guidelines set forth in "Fault Tree handbook"[6]. The known failure rates of sensors and the predicted failure rates of the HUD system were used along with predicted exposure times to determine, in accordance with the fault trees, the probability of occurrence of the hazardous events.

The System Failure Mode and Effect Analysis supported the Safety Analysis by establishing the failure rates and modes at the avionic unit level. The Detailed FMEA considered HUD System faults at the component level to support the system level analysis and to verify predicted built-in-test and monitoring effectiveness. In addition a failure test plan was prepared and tests were conducted to verify the detailed analysis.

6. Simulation and Flight Test Plans

A three axis motion simulator with visual scene was employed to develop the pilot system interface, evaluate pilot control response and to determine the ability of the system with the pilot-in-the-loop to meet touchdown footprint requirements in the specified environment. Prior to evaluating the system on the motion simulator the HUD Guidance System, airborne sensors, target aircraft aerodynamics, ground based ILS and environmental conditions were modeled and installed on a SEL 32/27 computer system. A pilot model[3] was also developed to permit complete modeling of the ILS capture, approach, flare and touchdown. All the sensor models and the ILS model included bias, gain variation and noise contamination. The environmental conditions modeled included steady winds, wind shears, turbulence, runway slope, runway length, glideslope angle, pressure, and temperature variations. Probability distributions for the identified disturbances were developed and used to generate random environments for the model. Aircraft weight, center of gravity and approach speed, as well as the pilot model were adjusted in Monte Carlo fashion for statistical evaluation of the system design.

These sensor errors, environmental conditions, aircraft configurations and their probability distributions were then applied to the manned

simulator to establish performance correlation between the pilot model and a large number of subject pilots. A manned simulation test program was established to demonstrate system performance in the CAT III environment. The pilot group consisted of airline pilots and FAA pilots. Pilot variability proved to be a large contributor to touchdown dispersions. The FAA demonstration program in the manned simulator required over 1000 flights to touchdown.

A flight test plan was then prepared to validate the fidelity of the simulation by demonstrating that the touchdown statistics from the simulation and the flight test were from the same population. The flight test plan required verifying performance in a subset of the environmental conditions used in the simulator demonstrations.

7. Selection of Flight Test Crew

To conduct the extensive flight testing of this experimental system in a large turbojet aircraft with the special requirement that some of the testing be done in actual CAT III weather, we developed special criteria for the selection of the crew and especially the chief pilot. The following requirements were applied in this selection.

1. Previous experience in CAT III operations
2. Previous experience in CAT III weather
3. Previous R and D flight testing
4. Ability to contribute to the development of operations procedures
5. Ability to act as a right seat safety pilot

We were fortunate to find an individual who could fill these roles well. We also received assistance from the FAA test pilots in a number of areas which helped make these tests successful.

8. Development and Installation of the Flight Data Acquisition and Touchdown Verification Equipment

The performance of the system in flight and at touchdown was gathered by a data acquisition system which received all the sensor data used by the HUD System, and stored this information on a high speed magnetic tape. An extensive data reduction system was then developed for the SEL 32/27 which provided time histories of sensor data, parameter/parameter plots and histograms across the flight test program. The system proved invaluable in determining aircraft and system performance, developing the final control laws, and in establishing the simulator flight test correlation.

A belly mounted video camera was employed to verify aircraft touchdown position. With a wide angle lens and proper calibration, touchdown position could be established longitudinally within 20 feet and laterally within 2 feet.

HUD symbology was continuously recorded on a second video system and overlayed with video from a forward looking camera mounted just behind the windshield. This video system permitted the flight test engineers to view the approach as if they were sitting behind the HUD.

9. Aircraft Operation

Due to the length of time involved from the beginning of CAT II flight tests to the completion of CAT III flights and the rising demand for 727 aircraft, two HUD system and data acquisition installations were required on two separate aircraft. Uninterrupted operation of the aircraft proved difficult as we were low in the priorities of the maintenance organizations willing to assist a one-of-a-kind aircraft traveling the Northwest in search of the winds and weather needed to satisfy the flight test plan requirements. Because all aircraft landing and take-off operations align for the best wind advantage, and our mission was to explore adverse winds, we were unable to take advantage of numerous opportunities at Type II airports since this would mean working against the traffic flow. Air Traffic Control did accommodate our operations whenever possible permitting the completion of the flight tests.

10. Simulation and Flight Test Data Confirmed Proof of Concept Test Results:

The simulation and flight test programs confirmed that a man-in-the-loop control system, subjected to the environmental requirements for CAT III operations can meet, and in this case, exceed the requirements for autoland systems. Data plots for the simulator and flight test results are presented in Figures 6 and 7. Most of the simulation and flight test landings were made with the pilots vision of the outside world obscured.

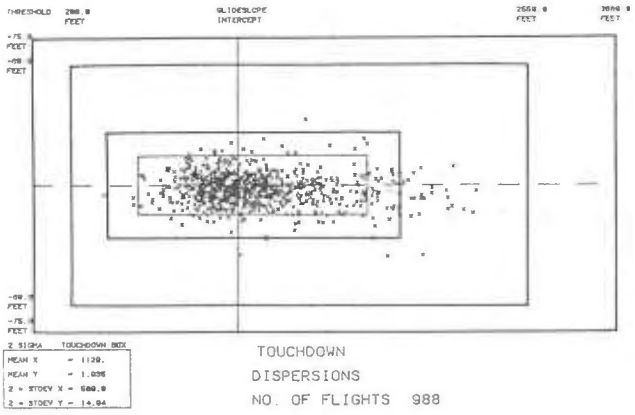


FIGURE 6
SIMULATION DISPERSIONS

Classical statistical hypothosis testing techniques were used to verify that the system performance dispersions met criteria set forth in AC 20-57A and AC 120-28C. Distribution fitting and extrapolation techniques combined with probability models of system monitor effectiveness were used for verification of low probability event statistics. Simulation statistics were shown to be reliable by showing statistical equality of flight test results and simulator results.

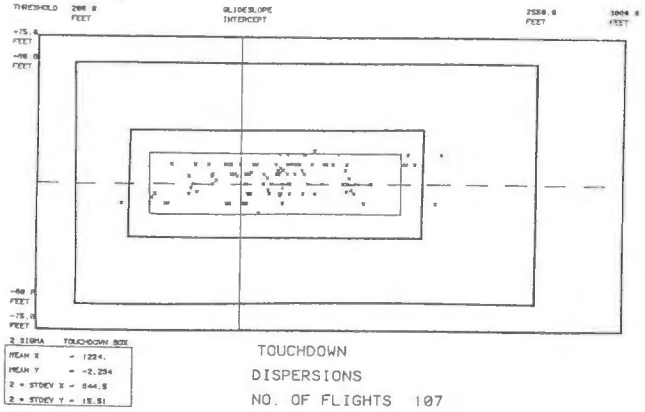


FIGURE 7

FLIGHT TEST DISPERSIONS

Conclusions

The ability of the HUD System to meet the requirements of CAT III operations has been examined: 1) in Proof-of-Concept testing where the effectiveness of single HUD pilot in-the-loop operations was demonstrated; 2) in simulation where performance and limit requirements were tested; 3) in the flight test aircraft where actual operations and correlation with simulation were demonstrated; and 4) in the engineering laboratory, simulation and aircraft where system failure modes have been analyzed, tested, and demonstrated. The issuance of the STC waits final review of reports and data and a test flight in actual weather conditions. This flight is scheduled for November of this year, 1984, when low visibility conditions occur on the U.S. West Coast.

The success of this program is credited to the assiduous design, development and test effort put forth by the staff at FDI, the assistance of our knowledgeable consultants and able flight crew, and the cooperation and support of the dedicated people of the FAA.

(1) "Criteria For Approval Of Category III Landing Weather Minima", FAA Advisory Circular Number: 120-28C
(2) "Automatic Landing Systems (ALS)", FAA Advisory Circular Number: 20-57A
(3) D.T. McRuer and E. S. Krendel, "Mathematical Models of Human Pilot Behavior", AGARD-AG-188: Jan., 1984
(4) McRuer, Jex, Clement, Graham, "A Systems Analysis Theory for Displays in Manual Control", STI Tech Rpt No. 163: June 1968
(5) McRuer, Ashkenas, Graham, "Aircraft Dynamics and Automatic Control", Princeton University Press: 1973
(6) Roberts, Vesely, Haasl, Goldberg, "Fault Tree Handbook", NUREG-0492: Jan., 1981

SESSION 15 COMMUNICATION, NAVIGATION, AND IDENTIFICATION (CNI) TERMINALS

Chairmen:

Dr. Duncan B.Cox, Jr.
Charles Stark Draper Lab, Inc.

Frank W. Smead
ITT Avionics

This session is primarily concerned with technical descriptions of actual or planned on-aircraft CNI terminals and with the issues and benefits associated with integrating several CNI functions together in one terminal.

Pages 447-453 have been deleted intentionally.

AN EXPERIMENTAL AERONAUTICAL SATELLITE DATA LINK

Samuel Anderson

Technical Staff
The MITRE Corporation
1820 Dolley Madison Boulevard
McLean, Virginia 22102

Abstract

There is a need to improve communication and surveillance capabilities in oceanic airspace. In October 1982, The MITRE Corporation initiated a 3-year, internally-funded project to design, develop, and demonstrate an experimental aeronautical satellite data link system to provide data communications capability between the ground and aircraft flying in oceanic airspace.

This paper describes the approach used by MITRE in developing the system and presents technical details to support specific design decisions. The experimental aircraft data terminal is designed to provide low-capacity data transfer between the aircraft and an earth station via satellite. The low aircraft terminal data rate of 200 bits per second (bps) minimizes power requirements and thereby the costs. The link between the aircraft and satellite is at L-band (1.5/1.6 GHz), and the links between satellite and earth stations are at C-band (4/6 GHz).

The subsystems comprising the aircraft data terminal include the transceiver, modem, processor, frequency control, and power. Digital logic functions are designed using STD Bus technology. The antenna design takes into account performance and cost tradeoffs between antenna gain and signal power.

The system concept avoids the high cost of launching, operating, and maintaining dedicated satellites through sharing of existing commercially available satellite links (e.g., INMARSAT) with other mobile users (e.g., maritime). The diversity among aeronautical users relative to their operational requirements and financial resources is accommodated by adopting a modular design architecture for user equipment that allows for expansion of capabilities at user discretion.

Background

The use of satellite channels has been considered since the early 1960s as an effective means of providing communications to a large group of aeronautical users. For some applications, such as communicating to and from aircraft flying remote routes, out of the range of ground-based VHF stations, satellite communication represents an ideal solution. Three satellites, correctly positioned, could conceivably provide coverage for all of the world's major air routes. The aeronautical satellite system concepts that have been studied to date, despite their theoretical and practical advantages, have been regarded as too expensive to be supported by aviation users, and none have been implemented. The cost of a satellite system dedicated to aeronautical users would involve the design and launch of satellites, earth station construction, maintenance, and technical support.

The economic consensus regarding the high cost of dedicated satellite systems is clear, yet there remains a problem of inadequate aeronautical communications over oceanic routes. Although the sophistication of on-board aircraft navigation systems is increasing, there is as yet no reliable means of conveying the data produced by these systems to the ground (dependent surveillance) when the aircraft is beyond the range of radar and VHF radio links.

The MITRE Corporation has committed large resources to the demonstration of a low-cost aeronautical satellite data link, which will use a space segment of an already existing satellite system. The purpose is to show that aeronautical satellite communications need not be overwhelmingly expensive while being of great value to the aviation community.

The concept is based on analyses and conclusions contained in an FAA report titled: "Oceanic Area System Improvement Study (OASIS)."[1] An OASIS working group estimated that reduced minimum airspace requirements in the North Atlantic, made possible by an automatic dependent surveillance capability derived from the shared use of the International Maritime Satellite (INMARSAT) system, could generate an overall cost savings of from 90 to 164 million dollars in the time period between 1979 through 2005, depending on the specific implementation.[2] An attractive feature of an INMARSAT-based aeronautical data link system is that aviation usage of the service could expand in increments. Beginning with limited satellite capacity initially, enough to serve selected applications, the user investment need increase only as existing demand and requirements increase.

The MITRE experimental data link will consist of one aircraft terminal and one earth-station terminal, which will operate over a shared-space segment of the present INMARSAT system. The aircraft terminal has been designed to provide a two-way communications link between the aircraft and the satellite at the existing maritime L-band mobile satellite frequencies (1.5/1.6 GHz).* (See Figure 1.) Both hardware and software have been developed to provide for multiple access, modulation and coding, and antenna switching.

Design Considerations

Cost Factors

It is not likely that an aeronautical satellite

*When the aeronautical mobile satellite frequencies are made available in the subsequent generations of the INMARSAT system, the aircraft terminal frequencies would also change correspondingly. The power budget given in Table 2 is based on the second generation INMARSAT spacecraft.

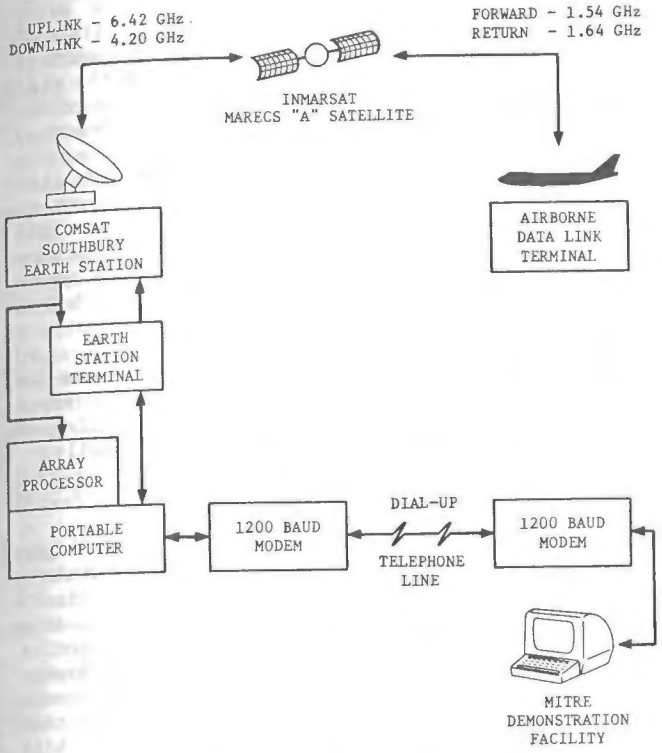


FIGURE 1
MITRE EXPERIMENTAL
AERONAUTICAL SATELLITE DATA LINK

data link design will be accepted unless it can be shown to be cost effective. This fact, when considered in a design perspective, places inevitable constraints on system performance. The cost of an implemented service would depend primarily on two parameters, described below, both of which have a first-order effect on the overall cost.

1. Earth Station Power. The subscription rates charged to users of a satellite system such as INMARSAT are based on the effective isotropic radiated power (EIRP) required from the satellite. For a given user channel, the satellite EIRP depends on the earth station transmitter power levels allocated to the specified channel and the gain provided by the satellite transponder. The aeronautical data link satellite EIRP requirement depends on several design variables such as data rate, aircraft antenna gain, and coding gain. Expected link signal and power parameters for the proposed MITRE concept are given in Tables 1 and 2 for the second generation INMARSAT spacecraft.[3-5]

MODULATION	DPSK
E_b/N_o (BER = 10^{-5})	10.4 dB
CONVOLUTIONAL CODING	1/2 rate, Constraint length-6
CODING IMPROVEMENT	3.5 dB
IMPLEMENTATION MARGIN	2.5 dB
DATA RATE	200 bps
REQUIRED CARRIER-TO-NOISE DENSITY RATIO (C/N_o)	32.4 dBHz

TABLE 1: SIGNAL PARAMETERS

	FORWARD	RETURN	
		HGA ON	HGA OFF*
TO SATELLITE			
FREQUENCY (GHz)	6.42	1.64	
TRANSMIT EIRP (dBW)	66.0**	11.3	17.7
FREE SPACE LOSS (dB)	200.9	189.2	189.2
PROPAGATION MARGIN (dB)	1.7	5.0#	5.0#
RECEIVE G/T (dB/K)	-14.0	-12.5	-12.5
C/N_o UP (dBHz)	78.0	33.2	39.6
FROM SATELLITE			
FREQUENCY (GHz)	1.54	4.20	
SATELLITE EIRP (dBW)	24.0**	-21.4	-28.0
FREE SPACE LOSS (dB)	188.5	197.3	197.3
PROPAGATION MARGIN (dB)	5.0#	2.0	2.0
RECEIVE G/T (dB/K)	-26.0##	32.0	32.0
C/N_o DOWN (dBHz)	33.1	39.9	33.3
C/N_o INTERMOD (dBHz)	66.5	70.0	70.0
C/N_o TOTAL (dBHz)	33.1	32.4	32.4

* The present generation MARECS-A satellite is equipped with a high-gain amplifier (HGA) provided primarily for maritime search and rescue applications. The next generation INMARSAT satellites will be equipped with a high-gain amplifier that will operate in a portion of the aeronautical band as well.

**Equivalent power of four INMARSAT maritime service voice channels.

\# Propagation margin includes fading, polarization, and absorbtion losses.

##Receive system noise temperature is assumed to be 500° K and the antenna gain is assumed to be 1.0 dB.

TABLE 2: LINK PARAMETERS AT 5° ELEVATION ANGLE

2. Aircraft Terminal Cost. The single aircraft-terminal cost should include hardware and software, installation, antenna(s), and maintenance costs.

Capability Tradeoffs

System Capability. The first compromise in the design of a low-cost data link is a restriction on channel capacity. The channel capacity of the system, or the achievable data rate, is a function of the satellite EIRP for both the ground-to-air (forward) and air-to-ground (return) links. Therefore, data rates must be restricted in order to (1) avoid high service charges for the forward link, (2) minimize the cost of the power amplifier in the aircraft terminal transmitter on the return link, and (3) minimize the cost and complexity of signal processing electronics. It is clear that unless there is a revolutionary breakthrough in digital voice encoding, the proposed data link will not have sufficient capacity to provide voice communications (1,200 to 32,000 bps).

The experimental data link has been designed to handle the data rates required for air traffic control (ATC) and automatic dependent surveillance functions as derived in the OASIS report. Automatic dependent surveillance of air traffic relies on navigational position data generated on board the aircraft and automatically communicated

to ATC ground stations. Independent surveillance describes surveillance methods which do not require such aircraft-generated data (e.g., radar). The average network data throughput requirement for North Atlantic air traffic through the year 2005 (223 in-flight aircraft) has been projected to be 200 bits per second (bps) over the forward link and 700 bps over the return link.[1] The return link capacity can be divided between four channels, each having a 200 bps capacity.[2]

In addition to lowering required signal power levels, the use of low data rates takes advantage of the availability and low cost of microelectronics, i.e., microprocessors that can handle most signal processing and encoding tasks when the data-rate-to-microprocessor-clock-rate ratio is low enough. It so happens that 200 bps approaches the upper data rate limit (400 bps may be achievable, but certainly nothing higher) for the type of microprocessor-based demodulator and error correction decoder implementations used in the experimental aircraft terminal design. (The experimental system operates at a standard data rate of 200 bps using 1/2 rate error-correction coding--2 channel symbols per information bit, or 400 channel symbols per second--with 8-bit, 4 MHz clock-rate microprocessor technology.)

The Aircraft Antenna

The aircraft antenna is a critical design feature because it plays a pivotal role in the data link cost tradeoff considerations. If, for a typical link power budget calculation, we assume a simple aircraft antenna with a hemispherical gain of +1 dBiC, the satellite L-band power required in the satellite-to-aircraft link is equivalent to approximately four INMARSAT maritime service voice channels.[2] On the other hand, if the aircraft antenna gain can be assumed to be somewhat higher, on the order of +6 dBiC, then the equivalent power of slightly more than one INMARSAT channel is required. Another significant advantage of a higher gain antenna is that because the antenna's directional discrimination is greater, channel degradation due to multipath interference is reduced. Probably the most sophisticated and best antenna for this application is a high-gain steerable phased array; however, a phased array does not represent an option in the aeronautical data link design being considered here because such antennas are not currently available for what would be considered an acceptable cost.

Under the imposed cost constraints, the optimum antenna configuration appears to consist of two microstrip antenna elements, flush-mounted on each side of the aircraft, approximately 40° down from top-center. Each antenna element provides a gain of at least +1 dBiC over a 130° spherically symmetrical angle, and much better gains at angles approaching vertical. Their optimum peak gain in the vertical direction, relative to the plane of the antenna element, is about +5 dBiC. For over-ocean communications in the North Atlantic, the dual, side-mounted antenna configuration should be able to exploit some of this directivity because the one antenna element in use at any given time will be pointed in the general direction of the INMARSAT satellite for typical aircraft orientations and flight paths in the North Atlantic flight track system.

The Aeronautical Multipath Channel

The Multipath Problem. The most complex technical problem confronting the system design is the signal degradation caused by the aeronautical channel's multipath characteristic. A simple description of multipath interference follows. When the direct-path satellite signal arrives at the aircraft antenna, it is accompanied by an assortment of reflected signals, varying in phase and intensity. Out-of-phase reflections add destructively to the direct-path signal and cause fades whose durations and magnitudes vary randomly. The resulting interference to the desired signal is referred to as multipath interference. It poses a difficult problem because it cannot be overcome entirely by increasing transmitter power levels since the intensities of the reflected path signals would also increase proportionally.

Overcoming Multipath Through Coding

Error Correction Codes. Aside from using a high-gain antenna with good directional discrimination, another way to contend with the multipath problem is by the use of error correction coding with interleaving. The theory of error correction codes is an extension of the well-known single-error-detecting parity check coding techniques. One significant difference in error-correction coding is that the quantity of redundant bits transmitted with the encoded message may vary from a small to a quite large percentage of the total message depending on the code rate used. The redundancy is exploited by the decoding algorithm to recover the original message from a received noise-corrupted encoded message if the number of errors in the received message is less than some maximum defined by the code.

Interleaving. Most error-correction codes are efficient at correcting random errors; however, in a multipath environment, the errors caused by fading may extend over long groups of encoded symbols, called error bursts. To manage the decoding of a message with error bursts, interleaving is used. It is a technique that essentially randomizes the channel errors received by the error correction decoder. It is performed in the transmitter processor after the data has been encoded and prior to transmission over the channel. A block of encoded symbols are reordered such that symbols ordered consecutively before interleaving are interspersed in the channel symbol stream, separated by a number (equal to the interleaving depth) of similarly reordered symbols. At the receiver, after the deinterleaving process is completed, channel burst errors appear to the error correction decoder as random, uncorrelated errors.

Interleaved Convolutional Encoding/Viterbi Decoding. The error-correction code selected for the experimental aeronautical data link is a constraint-length-6, convolutional code; the decoder is a software-implemented soft-decision Viterbi decoding algorithm. The interleaving/deinterleaving system uses an interleaving depth of 16 symbols. This value was arrived at as a compromise between a sufficiently long interleaving depth to decorrelate channel interference and a reasonably short minimum message length. Using this interleaving depth and a block interleaver, the shortest allowable data link message is 128 bits. While for most applications this is

quite short, air traffic service applications generally would not require messages longer than 40 bits.[1]

The complexity of the Viterbi algorithm increases exponentially with the constraint length. The error correction capability of the convolutional code also increases with constraint length; in a Gaussian noise channel, a constraint-length increment of one provides an additional link performance gain of approximately 0.5 dB.[6] The experimental data link's convolutional code uses a constraint length of 6, which is less than desired, but the best possible under the real-time constraints imposed by the microprocessor. Typically, convolutional codes, used with Viterbi decoders in satellite data links, have constraint lengths ranging from 7 to 10.

Soft-decision Decoding. One of the reasons the Viterbi decoding algorithm was selected is that it is capable of processing soft-decision data very easily. A demodulator which performs channel measurements with resolutions higher than just the common two-state, one/zero measurement is said to make soft decisions. If the demodulator performs 3-bit, 8-level A-to-D conversions, a performance improvement of up to 2 dB over typical 1-bit conversions can be provided when soft-decision decoding is used.[6]

Reliability Requirements

As yet, no reliability requirements have been established for the various potential aeronautical data link applications. For the experimental data link design, it has been assumed that the probability of a channel bit error should not exceed 10^{-5}.

Modulation and Demodulation

The Selected Modulation Technique. The modulation technique chosen for the experimental data link is differentially coherent phase-shift-keying (DPSK). This selection was based on a variety of criteria. Three of these criteria were: (1) power efficiency, (2) robustness in multipath fading, and (3) simplicity.

Although bandwidth constraints do not present a serious design restriction since data rates are much lower than the available bandwidth, spread-spectrum modulation techniques were not considered appropriate for this application. Because of the ample bandwidth, there was no point in considering M-ary PSK techniques, common in commercial communication applications which require high bandwidth efficiency. Likewise, multiple frequency shift keying (MFSK), as well as other technically viable modulation techniques such as minimal-shift keying (MSK), also were not considered seriously mainly because their implementations are significantly more complex than that of DPSK.[7] The selection alternatives were restricted to binary modulation techniques.

In an additive white Gaussian noise (AWGN) channel, phase-shift keyed (PSK) signals can be demodulated with a power efficiency 3 dB greater than frequency-shift keyed (FSK) signals. Coherent modulation methods, such as coherent binary PSK (BPSK), under the same AWGN channel conditions, perform slightly better than DPSK, a noncoherent technique. However, for this application, modulation performances must be evaluated in the aeronautical channel subject to phase noise and multipath fading.

The Phase Noise Problem. The low-data-rate data link application presents a design problem not encountered in higher capacity systems. Phase noise generated by oscillators in the earth station and aircraft terminals, as well as in the satellite, seriously degrades the performance of PSK systems at low data rates. Coherent demodulation systems are more seriously affected by phase noise than are noncoherent systems, due to the degraded performance of the integral phase-tracking loop necessary in coherent systems. According to calculations performed using INMARSAT phase noise specifications and the assumption of a slow fading channel, the performance of BPSK becomes unacceptable at channel symbol rates below about 350 bps, whereas DPSK is shown to perform adequately for symbol rates down to about 100 bps.[8] FSK systems are the most resistant to phase noise, but this does not outweigh their power efficiency disadvantage in AWGN channels.

Modem Performance in Multipath Fading. FSK modulation performs better than DPSK when severe fading conditions are present. However, in a standard nonfading environment, it performs 3 dB worse; and when coding and interleaving are used, FSK loses all of its advantages.[9]

Multiple Access

In a full-scale data link design, the multiple access system must accommodate as many as 223 aircraft that may be using the data link system at any one time. In the ground-to-air direction, the total channel capacity requirement (200 bps) is small enough to allow all users to share a common time-division multiple access (TDMA) channel. A discrete, address-mode service would operate by including an aircraft address with each message transmission. Each aircraft terminal would have the capability of recognizing its own address, whereupon it would respond appropriately.

The complexity in multiple access design is concentrated in the air-to-ground link. First, it seems probable that the 700 bps total required air-to-ground channel capacity would be split between four separate 200 bps channels, to form a time/frequency division multiple access (TDMA/FDMA) hybrid--essentially four separate TDMA channels. For a TDMA channel to operate in this multi-user aeronautical scenario, air-to-ground transmissions would either occur in response to ground-to-air polls or be very carefully timed. The earth station terminal must acquire the signal carrier (the frequency of which is unknown due to Doppler and oscillator uncertainties) for each air-to-ground message transmission. This carrier acquisition must proceed quickly for the system to achieve the necessary system throughput performance. The experimental data link will demonstrate the use of an array-processor fast Fourier transform technique to quickly estimate the carrier frequency for fast carrier acquisition on air-to-ground transmissions.

Additional complexity is associated with air-to-ground message timing and arbitration. Each aircraft terminal must be synchronized to the earth

station terminal to ensure that: (1) more than one aircraft message does not crowd the channel at any point in time; and (2) that the earth station terminal is in its carrier acquisition mode during the time any aircraft terminal may be sending its carrier-burst message preamble to initiate carrier synchronization.

Message Format

Appropriate message format structures are required for the data link design. However, since at this time no specialized services have been proposed, it would be premature to try to define them. The message formats that will be used with the experimental aeronautical data link will closely resemble typical INMARSAT message formats.

The message formats (illustrated in Figure 2) must contain: (1) a preamble consisting of a carrier recovery field (in return direction only), a bit timing field, and a "unique word" for frame synchronization; and (2) an information field which will contain the aircraft address, the message type, and the data field.

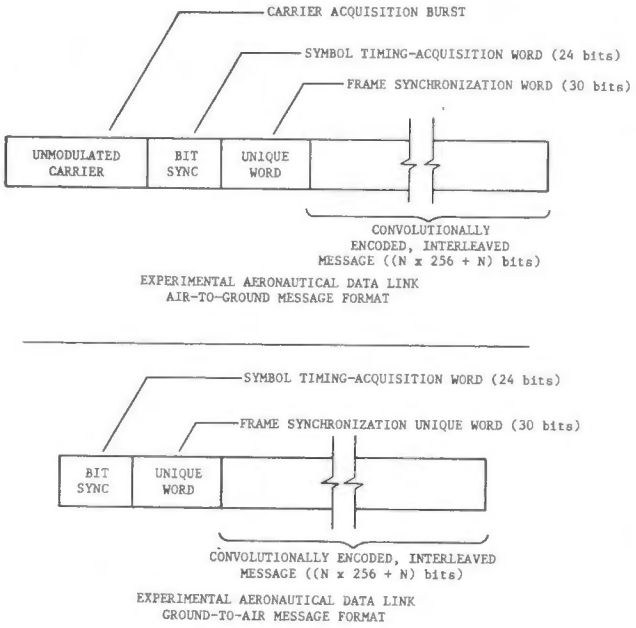


FIGURE 2
MESSAGE FORMATS

Aircraft Terminal Design Configuration

This section provides a description of the experimental aircraft terminal that will be used to demonstrate and test the aeronautical data link concept. The aircraft terminal design may be divided into two major parts—(1) the analog, RF section; and (2) the digital section. Figure 3 shows a schematic diagram of the principal aircraft terminal subsystems. The power subsystem is associated with both the RF and digital sections. A major design objective has been to develop a simple system that can be built from low-cost, off-the-shelf modules. The emphasis is on low-risk, commonly available technology, which minimizes the need for circuit-level hardware design.

RF Section

The RF section (Figure 4) consists of four basic subsystems:

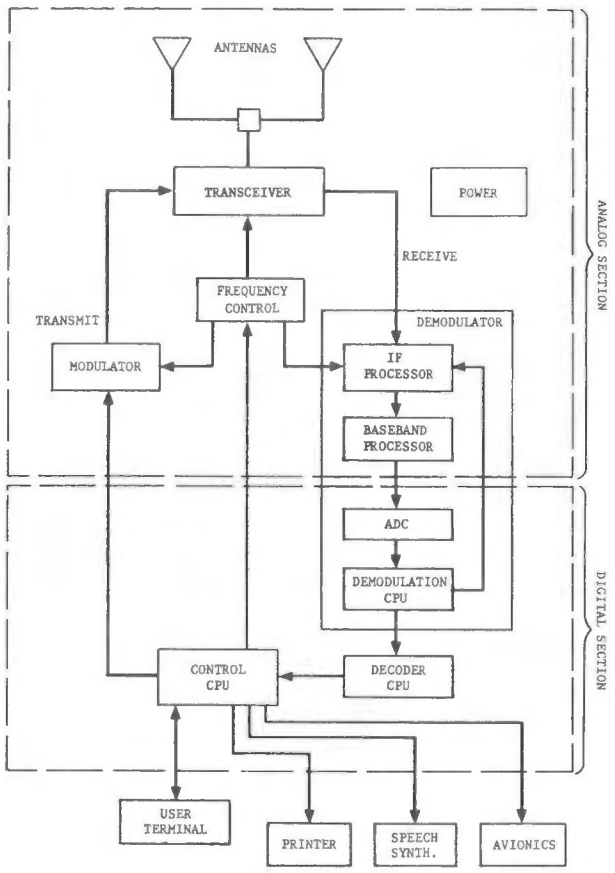


FIGURE 3
AIRCRAFT TERMINAL BLOCK DIAGRAM

1. Antenna Subsystem
2. Transceiver
3. Channel Selection Subsystem
4. IF Subsystem

Antenna Subsystem. The antenna subsystem consists of the following components:

1. Antennas
2. Coaxial cable
3. RF switch

The optimum antenna configuration, in terms of a compromise between cost and performance, is a pair of two side-mounted microstrip antennas, positioned 40 degrees down from vertical on each side of the aircraft. It is hoped that side-mounting two microstrip elements will optimize the antenna directivity relative to the satellite position for relatively low elevation angles so that coverage is available even during banks and turns.

The drag produced by antennas is a critical cost factor to high-performance aircraft. The microstrip elements are optimum in this respect since they are small and less than 0.10 inch thick; therefore, drag is minimized.

The length of coaxial cable required to connect the antenna to the aircraft terminal transceiver

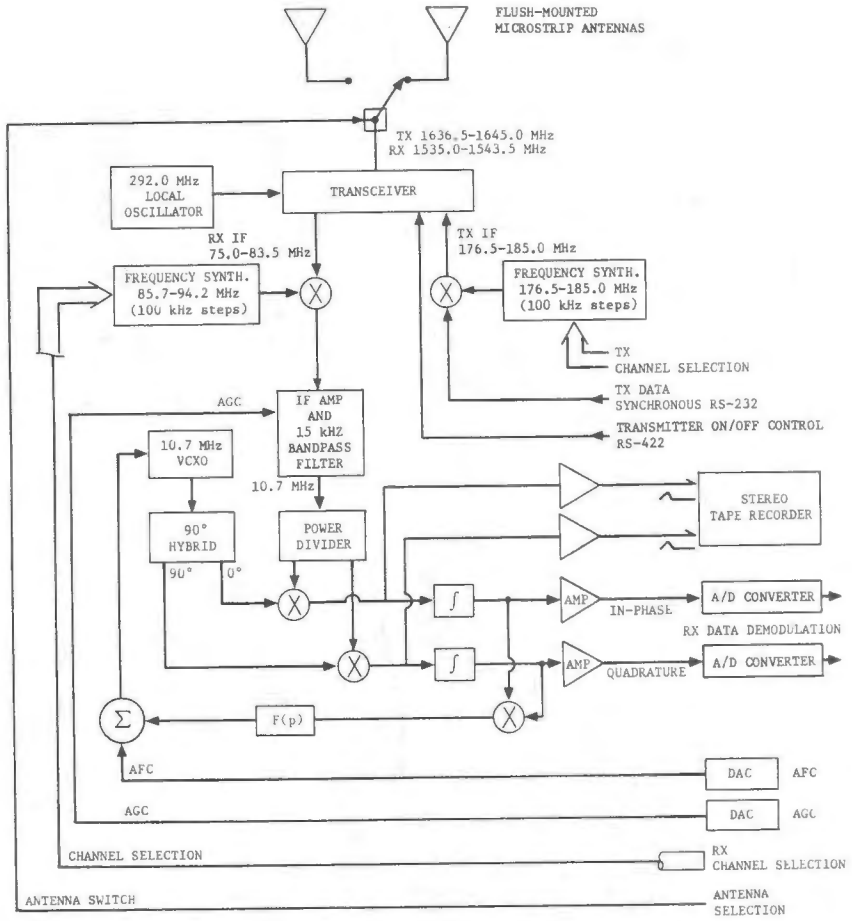


FIGURE 4
AIRCRAFT TERMINAL ANALOG SECTION

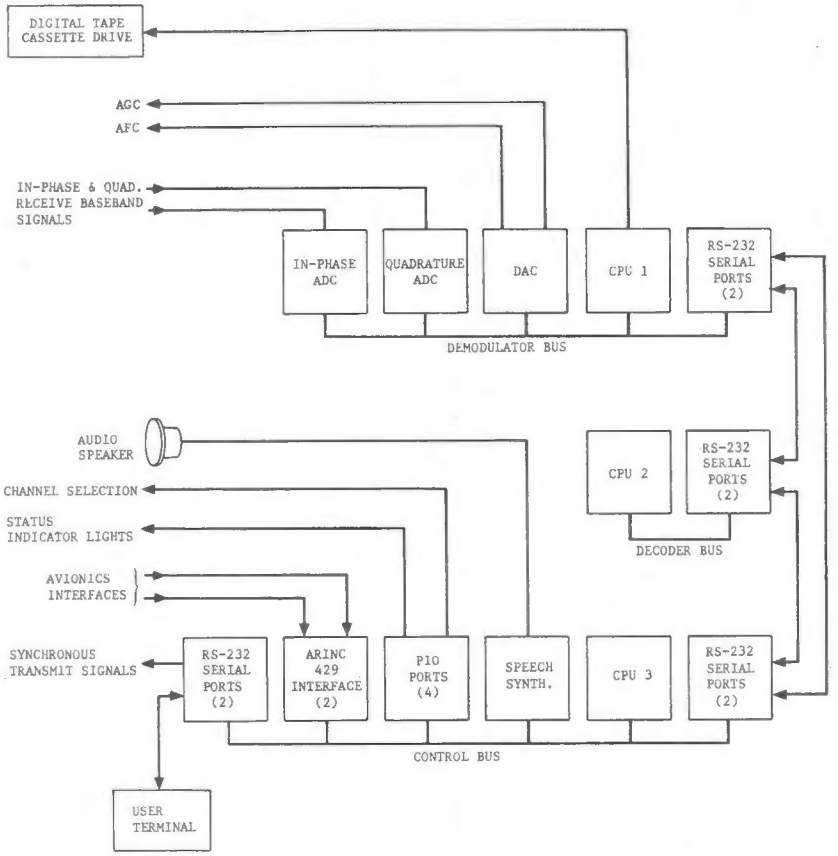


FIGURE 5
AIRCRAFT TERMINAL DIGITAL SUBSECTION

terminal transceiver is another critical design issue. At L-band frequencies, low-loss coaxial cable causes an attenuation of approximately 3 dB per 100 ft. Because of this, it is desirable to mount the antennas as close as possible to the equipment bay, where the aircraft terminal would be located. Another alternative is to mount the L-band transceiver somewhere close to the antenna and connect the IF and signal processing electronics to the transceiver via coaxial cable. Attenuation losses are less severe at IF frequencies.

The signals from only one of the two microstrip antennas will be received at any time, depending on which antenna is pointed most directly toward the satellite. Under processor control, an electrically controlled RF coaxial switch will select the proper antenna.

Transceiver The transceiver used in the experimental aircraft terminal design is a standard INMARSAT transceiver. L-band transmit and receive frequencies are 1636.5-1645.0 MHz and 1535.0-1543.5 MHz respectively. The transmit IF input is 176.5-185.0 MHz; the receive IF output is 75.0-83.5 MHz. The specified noise figure of the receiver is 2.5 dB under all conditions. The maximum power output of the transmitter is 28 watts.

A diplexer allows the use of a single RF antenna port to support both the transmitter and receiver.

Channel Selection Subsystem. The channel selection subsystem consists of a pair of frequency synthesizers, digitally controlled, that operate in the transmit and receive IF frequency bands to enable transmit and receive channel selection. Frequency synthesizers with the high performance specifications for low phase and low spurious noise required by this design are generally not off-the-shelf items. Therefore, the synthesizers used to build the aircraft terminal have frequency steps of 100 kHz, larger than desired by a factor of four.

IF Subsystem. The receive channel frequency synthesizer will downconvert the receive channel transceiver IF output to 10.7 MHz. The demodulation processing of the received signal begins with a 15 kHz bandpass filter centered at 10.7 MHz. The 10.7 MHz band signal is passed through a power divider; the two outputs are then mixed with voltage controlled oscillator (VCXO) frequencies, 90 degrees out of phase, to produce in-phase and quadrature baseband outputs. The VCXO is centered at 10.7 MHz but is variable under processor control to facilitate the automatic frequency tracking function.

The in-phase and quadrature baseband signals pass through an analog signal processing section consisting of an integrator and an amplifier, after which the signals are sampled by an analog-to-digital (A/D) converter for further processing in the digital section.

The Digital Section

The aircraft terminal digital hardware subsystem, illustrated in Figure 5, consists entirely of 8-bit Standard Bus (STD Bus) modules. Three processors (STD Bus CPU modules) operate on three separate busses; no bus arbitration is available with STD Bus technology to manage shared-bus systems. This arrangement is effective because the processors handle three almost independent tasks. The processing load is divided evenly between them, and interbus, interprocessor communications are handled with sufficient efficiency via asynchronous serial communications ports.

The three processors are referred to as follows:

1. Demodulation Processor
2. Decoder Processor
3. Control Processor

Demodulation Processor. The demodulation processor operates on a bus in conjunction with four other STD Bus modules:

1. two analog-to-digital converters (ADCs) used to sample the in-phase and quadrature baseband signals;

2. one digital-to-analog converter (DAC) module with multiple outputs used to control automatic gain and frequency tracking loops; and

3. one serial communications module with two ports, i.e., one port for interbus communications and one for a digital tape cassette interface.

The demodulation processor handles the real-time sampling, bit synchronization, demodulation, and signal control functions for the aircraft terminal. When a DPSK demodulation decision is formed, the 3-bit soft-decision value is passed to the decoder processor.

Decoder Processor. The decoder processor operates on a bus with just one other module: a serial communications module to provide interbus communications to the other processors.

The functions of the decoder processor are frame synchronization; code-block formatting; and buffering, deinterleaving, and decoding. When a message has been completely decoded, it is passed to the control processor.

Control Processor. The control processor operates on a bus with four other modules:

1. two serial communications modules to provide interbus communications, a user terminal interface, and synchronous data link transmit control;

2. one ARINC 429 (airline industry digital bus standard) dual-port serial avionics interface module; and

3. one speech synthesis module.

The main control processor functions are (1) to provide a user terminal monitor with simple menu and editing features; (2) to perform all data link transmit functions, such as message formatting, interleaving, convolutional error-correction encoding, differential encoding, and synchronous manchester encoded transmission; (3) to provide software arbitration for all digital interfaces, such as those to avionics systems and test equip-

ment; and (4) to act as a message processor which will operate on ground-to-aircraft messages and perform appropriate functions according to message type. Unlike the other processors, the control processor does not have a dedicated real-time-sensitive function. It is flexible in the sense that much of its software may be written in a high-level language, and can therefore be easily augmented, updated, and modified.

Test and Demonstration Plans

The aeronautical data link will be flight tested in the summer of 1985 using an INMARSAT satellite channel. The objectives of the tests will be to prove the overall concept and to generate data that will be of use in developing engineering refinements for a potential future full-scale data link design. Some objectives of the demonstration program are as follows.

Demonstration of Remote Automatic Dependent Surveillance

As concluded in the OASIS final report, and stated earlier in this paper, the data link capacities required for automatic dependent surveillance are on the order of 200 bps per channel for the communication of navigational data generated by on-board navigation systems to air traffic control centers, so that position tracking of en route aircraft can be maintained.

For the flight tests, the MITRE aircraft terminal will be equipped with an avionics interface to receive data from a standard navigation unit; the data will be periodically transmitted to the ground segment. For demonstration purposes, the data may be communicated from the earth station in Southbury, Connecticut to a MITRE-based demonstration site where a computer graphics display will show the progress of the tracked aircraft.

Channel Verification Tests

The next order of priority will be the channel verification tests. Bit error rate (BER) tests will be performed at three transmit power levels. Channel propagation tests will also be performed at three power levels to measure the channel multipath characteristics as well as to measure carrier acquisition performance levels. The tests will be performed while flying at a geographical latitude that will ensure that the test results are representative of worst-case satellite elevation angles in the North Atlantic organized track system, but not necessarily the polar routes.

Demonstration of Message Transfer

Basic message transfer capabilities will be demonstrated. This capability forms the basis of a number of possible user services, such as company business data communications and telex.

Demonstration of Weather Data Transfer

Based on previous work done at MITRE, it should be possible to demonstrate the ground-to-aircraft transfer of simple weather map facsimiles.

Summary

The design of an experimental aeronautical satellite data link has been accomplished using off-the-shelf components and a simple configuration of three microcomputers. The main objectives of this project are to determine the required level of satellite power necessary to operate a low-capacity aeronautical data link and to demonstrate the feasibility of an air-ground communications system for aviation applications which uses a shared satellite system. After the project flight tests have been completed at the end of fiscal year 1985, sufficient data should be available to enable an efficient full-scale design. Further details concerning the aircraft and ground station terminal designs, as well as signal processing and multiple access strategies, will be made available in the final project report.

Acknowledgments

Acknowledgments are given to Walter C. Scales who conceived and initiated this project and performed the systems studies and high-level design, and to Lawrence Hogle and Yaroslav Kaminsky, who helped significantly in the preparation of the article.

References

[1] G. J. Couluris and B. Conrad, "Oceanic Area System Improvement Study," SRI Report No. FAA-EM-81-17, Vol. 1, September 1981.

[2] OASIS Working Group B "Consideration of INMARSAT for an Aeronautical Data Link Only Service for the NAT," ARC 82/WP-5, WGB, November 1981.

[3] "Revised Characteristics of INMARSAT Second Generation Satellite Transponders," CCIR, IWP 8/7 Sub-group, Document No. INMARSAT/15, 19 October 1983.

[4] "Satellite EPIRB Coordinated Trials Programme Using the INMARSAT Geostatioary Space Segment Operating in the 1.6 GHz Band," CCIR, Interim Meeting of Study Group 8, Mobile Services, New Draft Report, Document 8/167, Table III; Geneva, May 1984.

[5] "Link Power Budgets for a Maritime Mobile Satellite Service," CCIR, XVth Plenary Assembly, Mobile Services Vol. VIII, Report 760-1, pp. 541-551, Geneva, 1982.

[6] J. A. Heller and I. M. Jacobs, "Viterbi Decoding for Satellite and Space Communication," IEEE Trans. Commun. Tech., Vol. COM-19 No. 5, October 1971.

[7] J. J. Spilker, "Digital Communications by Satellite," Englewood Cliffs, New Jersey, Prentice-Hall, 1977.

[8] S. Rhodes, W. Hagmann, P. Chang, and R. Fang, "Signal Design for INMARSAT Standard C Ship Earth Stations," Globecom '82, Vol. 3, Miami, November 29-December 2, 1982.

[9] D. Chase, "Digital Signal Design Concepts for a Time-Varying Rician Channel," IEEE Trans. Commun. Tech., Vol. COM-24, February 1976.

AN ADAPTIVE AIRBORNE VLF COMMUNICATIONS ARRAY CONCEPT

Stephen R. Schmitt
Naval Air Development Center
Warminster, Pa. 18974

Abstract

A digital signal processing algorithm based on the Kalman filter is shown to be suitable for use in an airborne adaptive low frequency communications array. The Kalman filter is used to adjust the weights in a weighted sum of antenna element outputs in order to minimize the mean square error between this sum and a replica of the transmitted waveform. The algorithm uses decision aiding in which bit decisions indicate the correct replica waveform used in estimation of the optimal weights. An impulsive noise suppression subroutine is included as part of the algorithm to reduce the effects of atmospheric noise.

Introduction

An important frequency band used for radio communications at very long ranges is the region of the RF spectrum below 100kHz covering the Extremely Low, Very Low, and Low Frequency bands. Study of radio propagation and noise in these bands has continued for many decades (1-8). At frequencies below about 100kHz the region between the earth's surface and the ionosphere acts much like a waveguide. The dominant propogation modes are those with the electric field oriented vertically (TM mode). This fact makes possible signal reception with magnetic dipole antenna arrays in which the gain of an antenna element varies with azimuthal direction. The problem under study is that of applying adaptive array techniques to a magnetic dipole array in order to suppress interference. At frequencies below 100kHz it is impractical to make use of phase differences between antenna elements since the wavelengths at these frequencies are very long; however, the effects of varying the gain patterns of several antenna elements can be exploited for adaptive spatial filtering.

Communication System Simulation Description

The digital computer simulation used in this study includes mathematical models of an analog modulator, additive continuous wave (CW) interference, additive impulsive atmospheric noise, two loop antenna elements each connected to a frequency converter which converts to baseband, and a digital filter which optimally combines the baseband waveforms to recover the transmitted data.

A commonly used waveform for communication in the frequency band below 100kHz is the minimum shift keying (MSK) waveform. This waveform can be formed from the linear sum of two modulated phase-shift-keying (PSK) waveforms. The MSK waveform can be demodulated by a pair of PSK matched filters. In the simulations used in this study, a PSK waveform is assumed. The algorithms used in the digital filtering part of the simulation are, therefore, quite similar to those which would be employed for the MSK waveform.

The analog modulator which is simulated produces a binary PSK waveform at modulation rate 400/sec with a carrier at 6.4kHz. Two orthogonal 8-chip codewords are used to represent binary data resulting in a data rate of 50 bits/sec. The resulting waveform is a direct sequence spread spectrum signal. In a real system the codewords may be encrypted by mod-2 addition with a binary keystream at rate 400/sec.

The interfering waveform is a tone at the carrier frequency of 6.4kHz. The amplitude of the interfering waveform is set considerably larger than the amplitude of the signal. The Signal-to-Interference (SIR) power ratio is adjusted by changing the amplitude of the signal resulting in the simulation of a relatively constant power output from the frequency converters. The phase difference between the interfering waveform and the signal carrier is set randomly for each run of the computer simulation.

The method of simulating atmospheric noise is intended to represent the physical nature of the environment in order to study how the digital filtering algorithm would perform in the presence of this type of noise. It is not necessary to develope a highly accurate simulation of atmospheric noise for this purpose. The model of atmospheric noise assumes several discrete impulse sources distributed uniformly over the surface of a sphere. The strength of an impulse is reduced according to Pierce's formula (9) to simulate the effects of distance on attenuating electromagnetic waves propogating within the earth-ionosphere spherical waveguide. Each source generates impulses randomly. The loop antenna model, Pierce's formula, and the range and bearing of each source from the receiver are used to compute the output of each antenna element in order to simulate the reception of atmospheric noise.

Using Pierce's formula, the attenuation of a signal is proportional to:

$$\exp[(-a \cdot d/R)/\sin(d/R)]$$

in which:

- d - distance along the spherical surface from the signal source to the receiver
- R - radius of the sphere

a - attenuation constant (2.56 at 30kHz, 0.456 at 3kHz, 2 in the simulation)

Distribution of noise sources uniformly on the surface of a sphere can be easily accomplished by use of the fact that the area on a sphere between two parallel planes cutting the sphere in linarly proportional to the distance between them. Then:

$$d = R \cdot \arccos[(R-h)/R]$$

in which h is the distance on a line from the receiver through the center of the sphere. Letting h vary uniformly over [0,2R] and the azimuth angle vary uniformly over $[0,2\pi]$ allows the selection of pairs of points in terms of range and azimuth from a receiver which describe a uniform distribution of impulsive noise sources on the surface of a sphere representing the earth. The locations of noise sources can be set randomly for each run of the simulation. A statistical parameter, V_d, is used as a measure of atmospheric noise impulsivity (5). It is defined as the ratio of the root-mean-square to average voltage of the noise envelope at the output of a band pass filter. The atmospheric noise generation subroutine produces about 100 pulses per second from all sources which generally results in a value for V_d of about 10 dB in a bandwidth of 200 Hz.

The fact that the dominant signal propagation modes below 100kHz are those with the electric field vertical and the magnetic field horizontal is used in the mathematical model of a two element magnetic dipole antenna array. For a time varying magnetic field, B(t), the output of a single loop is:

$$v(t) = - A \cdot \cos(\theta) \cdot \dot{B}(t)$$

in which:

B(t) - magnetic induction, volt-seconds/meter^2

A - surface area of loop, meter^2

θ - angle between $\underline{B}(t)$ and a line perpendicular to the loop plane

The antenna array model used in this study consists of two magnetic dipoles with both axes in a horizontal plane and with both axes at right angles. The horizontal plane moves with aircraft pitch, roll, and yaw so that the antenna element ouputs are given by:

$$v_1(t) = g_1 s(t) \cdot \sin(a) \cdot \cos(b)$$

$$v_2(t) = g_2 s(t) \cdot \cos(a) \cdot \cos(c)$$

where:

g_i - antenna gain constants

s(t) - signal

a - angle of arrival of signal

b - aircraft pitch angle

c - aircraft roll angle

In simulating the antenna array, a sine wave signal is assumed and the antenna element ouputs are simplified to:

$$v_1 = s \cdot \sin(a) \cdot \cos(b)$$

$$v_2 = s \cdot \cos(a) \cdot \cos(c)$$

Each frequency converter simulated in this study produces inphase and quadrature components of the baseband waveform by a single stage conversion of the signal received from an antenna element followed by a low pass filter for each component. It is assumed that a highly stable reference signal is available at the receiver. This technique, when combined with adaptive digital filtering, would eliminate the need for a phase locked loop.

The low pass filter used in the computer simulation has the discrete transfer function:

$$G(z) = \frac{.11990976z}{z^2 - 1.7254245z + .75865088}$$

which results in a sampled data low pass filter of second order having a cutoff frequency of 796 Hz when the sampling rate is 25,600 samples per second (the sampling rate used in the simulation of the analog portion of the system).

Samples of the baseband waveform are obtained from each frequency converter at a rate of 1600/sec. The weight adjustment algorithm uses bit decisions to determine the correct reference waveform to be used in the weight adjustment process. For data synchronization, several correlators are run in parallel, each with a different timing hypothesis. The correlation result with the least error determines the estimate of the transmitted data and indicates the best estimate of waveform timing.

Digital Filter Algorithm

The digital filter (10) has the goal of minimizing the mean-square-error (MSE):

$$E\, e^2 = E[d(k) - \underline{w}'\underline{x}(k)]^2$$

which is defined as the average squared error between discrete samples of a reference waveform d(k) and the weighted sum of a set of signal samples $\underline{x}(k)$ taken at the output of the analog portion of the array at time kT. Since the simulated system communicates with a series of binary data waveforms, the correct reference signal is then a replica of one of two possible transmitted signals:

$d_0(k)$, k:1,$\cdots$,N if 0 transmitted
$d_1(k)$, k:1,$\cdots$,N if 1 transmitted

Matched filtering, in which:

$$y(k) = \underline{w}'\underline{x}(k)$$

is compared with $d_0(k)$ and $d_1(k)$ is performed. The comparison having the least cumulative error indicates the correct reference signal and gives

an estimate of the transmitted data. Once the correct reference signal is known, the algorithm can calculate updated sets of weights $\underline{w}$.

The mathematical model of the dynamic behavior of the optimal array weights can be represented by:

$$\underline{w}_o(k+1) = A(k+1,k)\,\underline{w}_o(k) + \underline{u}(k)$$

in which:

$$E\ \underline{u}(k)\underline{u}'(k) = Q(k)$$

and in which the weight transition matrix defines how the optimal weights $\underline{w}_o$ change as the aircraft maneuvers. The noise vector $\underline{u}(k)$ accounts for errors in the mathematical model of the system. The system measurements can be represented by a noise corrupted version of the optimal array output:

$$d(k) = \underline{x}'(k)\underline{w}_o(k) + v(k)$$

in which:

$$E\ v(k)\cdot v(k) = r^2(k)$$

A filtering algorithm for weight determination can be defined by use of the Kalman filtering equations in which $\underline{w}$ is the state vector:

$$\underline{w}_p(k+1) = A(k+1,k)\underline{w}_f(k)$$

$$P_p(k+1) = A(k+1,k)P_f(k)A'(k+1,k) + Q(k)$$

$$K(k) = \frac{P_p(k+1)\underline{x}(k)}{\underline{x}'(k)P_p(k+1)\underline{x}(k) + r^2(k)}$$

$$P_f(k+1) = P_p(k+1) - K(k)\underline{x}'(k)P_p(k+1)$$

$$\underline{w}_f(k+1) = \underline{w}_p(k+1) + K(k)[d(k) - \underline{x}'(k)\underline{w}_p(k+1)]$$

These equations constitute a Kalman filtering algorithm for determining the optimum values of the elements of $\underline{w}$ in which:

$\underline{w}_p(k)$ - predicted weight vector

$\underline{w}_f(k)$ - filtered weight vector

$A(k+1,k)$ - weight transition matrix

$P_p(k)$ - predicted covariance matrix

$P_f(k)$ - filtered covariance matrix

$K(k)$ - Kalman gain matrix

The weight transition matrix may be derived for the case under study by making use of the optimal solution for the weight vector for the baseband waveform determined by four equations in four unknowns:

$$2\cos(Ps)/As = w_1\cdot g_1\cdot\sin(a_s)\cdot\cos(b) + w_3\cdot g_2\cdot\cos(a_s)\cdot\cos(c)$$

$$2\sin(Ps)/As = w_2\cdot g_1\cdot\sin(a_s)\cdot\cos(b) + w_4\cdot g_2\cdot\cos(a_s)\cdot\cos(c)$$

$$0 = w_1\cdot g_1\cdot\sin(a_j)\cdot\cos(b) + w_3\cdot g_2\cdot\cos(a_j)\cdot\cos(c)$$

$$0 = w_2\cdot g_1\cdot\sin(a_j)\cdot\cos(b) + w_4\cdot g_2\cdot\cos(a_j)\cdot\cos(c)$$

This can be written in the matrix form:

$$\underline{e} = S\underline{w}$$

The incremental changes in the optimum weight vector are determined by a weight transition matrix so that:

$$\underline{w}(k+1) = A(k+1,k)\underline{w}(k)$$

Since the phase, amplitude, and locations of the signal and interference sources are approximately constant:

$$\underline{e} = S(k)\underline{w}(k) = S(k+1)\underline{w}(k+1)$$

and,

$$\underline{w}(k+1) = S^{-1}(k+1)\underline{e}$$

then by substitution:

$$S^{-1}(k+1)\underline{e} = A(k+1,k)S^{-1}(k)\underline{e}$$

giving:

$$A(k+1,k) = S^{-1}(k+1)S(k)$$

The change in azimuth angle, roll, and pitch from time kT to (k+1)T are given by:

$$a_s(k+1) = a_s(k) - da(k+1)$$

$$a_j(k+1) = aj(k) - da(k+1)$$

$$b(k+1) = b(k) + db(k+1)$$

$$c(k+1) = c(k) + dc(k+1)$$

in which da, db, and dc are the incremental changes in heading, pitch, and roll of the aircraft. By performing many trigonometric and algebraic manipulations, the elements of the weight transistion matrix can be reduced to:

$$a_1 = \cos(da)\cdot\cos(db) + \cos(da)\cdot\sin(db)\cdot\sin(b)/\cos(b)$$

$$a_2 = -\sin(da)\cdot\cos(dc)\cdot\cos(c)/\cos(b) - \sin(da)\cdot\sin(dc)\cdot\sin(c)/\cos(b)$$

$$a_3 = \sin(da)\cdot\cos(db)\cdot\cos(b)/\cos(c) + \sin(da)\cdot\sin(db)\cdot\sin(b)/\cos(c)$$

$$a_4 = \cos(da)\cdot\cos(dc) + \cos(da)\cdot\sin(dc)\cdot\sin(c)/\cos(c)$$

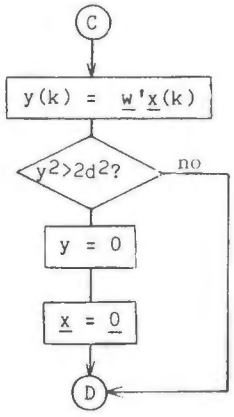
in which:

$$A(k+1,k) = \begin{bmatrix} a_1(k+1) & 0 & a_2(k+1) & 0 \\ 0 & a_1(k+1) & 0 & a_2(k+1) \\ a_3(k+1) & 0 & a_4(k+1) & 0 \\ 0 & a_3(k+1) & 0 & a_4(k+1) \end{bmatrix}$$

The terms of each trigonometrix expression are evaluated at time (k+1)T. It can be seen that the weight transition matrix may be computed using only measurements of aircraft heading, pitch, and roll.

The simulated algorithm updates the covariance matrix and the Kalman gains once each sample period. A pair of filtered and a pair of predicted weight vectors for each timing hypothesis is updated each sample period with each weight vector in a pair corresponding to one of two possible bit decisions. At the end of a bit period for each timing hypothesis a bit decision is made and stored. The associated filtered weight vector is retained as the starting filtered weight vectors for both possible bit decisions in the next cycle. After a complete set of timing hypothesis parameters have been updated, the bit decision corresponding to the best timing estimate is chosen as the filter output.

Impulsive atmospheric noise can affect an adaptive array algorithm adversely, causing momentary losses of the signal or preventing algorithm convergence in some cases. The effects of atmospheric noise can be mitigated to some extent if a "hole punching" subroutine is added to each adaptive array algorithm. Note that the adaptive process causes the average power of the array filter output to approach the average power of the reference waveform as the adaptive process continues period. This process is quite rapid. The hole punching subroutine performs the following algorithm:



This algorithm zeroes the current filter output and the input samples if a noise pulse is detected. A noise pulse is indicated in the hole punch subroutine if the instantaneous filter output power is two times or 3 dB greater than the estimated signal power. The hole puncher is designed to suppress large noise pulses which have the greatest adverse effects on the adaptive algorithms. In the Kalman filter, replacing each sample with zero prevents the update of the filtered weight vector and the filtered covariance matrix by causing the Kalman gain vector to be zero during the sample in which a noise impulse is removed. The bit decision correlators will also not be incremented for a large impulse to improve the bit decision error rate.

Simulation Results

Computer simulations were run in which the signal-to-noise power ratio was set to -30 dB for signal-to-interference power ratios of -20, -40, and -60 dB. The SNR of the filters output for each case are graphed in figures 1 - 3. The simulation program included a subroutine which modeled the effect of a rapid aircraft maneuver on antenna attitude. Six seconds of real time were simulated with straight and level flight for the first second, a roll to a bank angle of 45 degrees during the next second, and a turn at 9 degrees per second during the remaining four seconds.

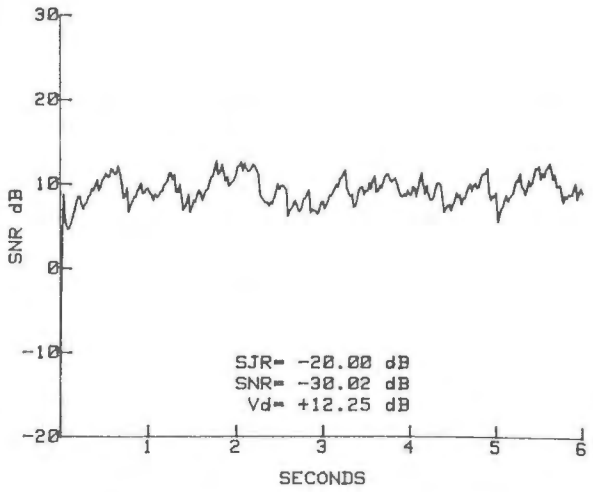


Fig. 1 Filter output SNR at -20 dB SIR

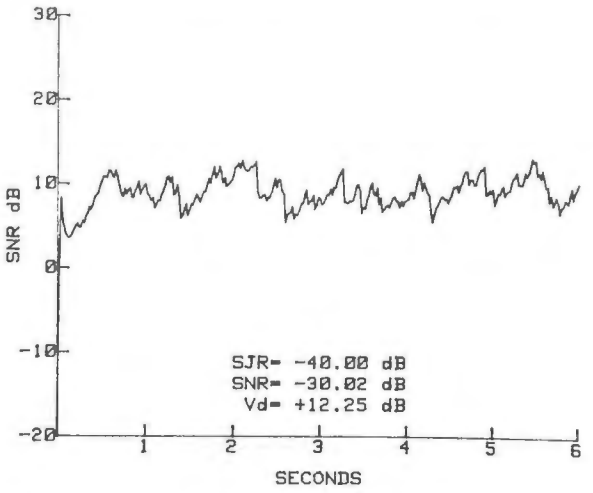


Fig. 2 Filter output SNR at -40 dB SIR

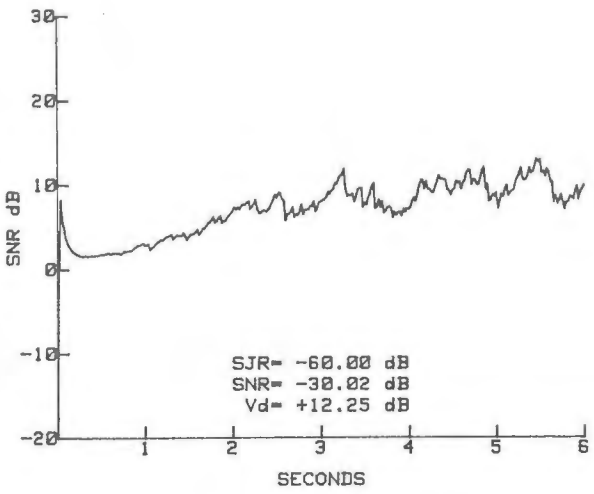


Fig. 3 Filter output SNR at -60 dB SIR

Conclusions

The result of the study suggest that relatively high data rate transmission can be maintained in low frequency communication systems in the presence of strong interference and high levels of atmospheric noise by using an adaptive array based on the Kalman filter. However, the computational burden imposed, assuming a 20 msec sync window for instantaneous data synchronization, is 4.7 million arithimetic operations per second. By using a search technique for data synchronizations the computational burden may be reduced to 557 thousand arithimetric operations per second. Practical implementation of this algorithm would require a compact processor capable of very high speed computation.

References

(1) Special Issue of the Proceedings of the IRE, Vol.45, No. 6, June 1957

(2) I. J. Rothmuller, "Effect of the VLF Propagation Channel on Spread -Spectrum Communications Systems," Naval Electronics Laboratory Center, San Diego, CA, NELC Technical Report 1834, 4 August 1972

(3) F. J. Kelly, J. P. Hauser, H. M. Beck, and F. J. Rhoads, "Multipath VLF Propagation Effects on Correlation Receivers," Naval Research Laboratory, Washington, DC, NRL Report 8521, 18 September 1981

(4) E. L. Maxwell and D. L. Stone, "Natural Noise Fields from 1cps to 100kc,"IEEE Transactions on Antennas and Propagation, pp.339-343, May 1963

(5) International Radio Consultative Committee, "World distribution and characteristics of atmospheric radio noise," International Tele-communications Union, Geneva, Switzerland, Report 322, 1964

(6) J. K. Omura and P. D. Shaft, "Modem Performance in VLF Atmospheric Noise, "IEEE Transactions on Communications Technology, Vol. com-19, pp. 659-668, 1971

(7) E. Bolton, "Simulating Atmospheric Radio Noise from Low Frequency Through High Frequency," The Review of Scientific Instruments, Vol. 42, No. 5, pp. 574-577, 1971

(8) A. Shohara and C. K. Un, "VLF/LF Modem Performance in Atmospheric Noise: Theoretical Analysis," Stanford Research Institute, Menlo Park, CA, SRI Project 4669, Vol. I, March 1977

(9) A. D. Watt, "VLF Radio Engineering," Pergamon Press, 1967, p.240

(10) R. A Monzingo and T. W. Miller, "Introduction to Adaptive Arrays," John Wiley and Sons, New York, 1980

SESSION 16 SYSTEMS AND SOFTWARE - ADA™

Chairman:

Dennis B. Mulcare
Lockheed-Georgia Co.

Tom McTigue
McDonnell Aircraft Co.

This session provides a wide ranging treatment of the Ada programming language as it applies to avionics applications; including mission and flight critical software, fault-tolerant operating systems, run-time support software, and actual experience.

DIRECTED GRAPH METHODOLOGY©

Elizabeth M. Lanier, Associate Engineer
Michael E. Hinkey, Engineer

Westinghouse Electric Corporation
Defense and Electronics Center
Systems Development Divisions
Baltimore, MD 21203

Abstract

Directed Graph Methodology (DGM) is an integrated set of design tools that assist in documentation, specification and translation of signal processing applications. DGM clarifies system requirements by constructing graphs out of signal processing block diagrams and through Ada® language constructs which facilitate translation of the application. Graphs are represented by nodes, or functions such as Ada packages, executing on processors; queues, representing the flow of data from processor to processor; and control variables, the means of communicating with any process outside of the graph itself, such as another graph or a switch on a console. Two pieces of software, the DGM Editor and the Library Manager, allow the user to create a directed graph in a controlled environment and automatically document his graph and interface with each of the other tools in the design system. The DGM Editor provides a link to a variety of simulators which verify data flow and algorithm execution and to a translator that creates the necessary data structures for a multiprocessing operating system. Future work in Directed Graph Methodology includes the use of Artificial Intelligence to incorporate each tool into an expert system and further automate the design cycle. DGM allows the user to more easily define hardware and software requirements and save time in both the design and integration of systems software. The use of simulation techniques throughout the design phase ensures that the application code is correct and that the hardware is capable of meeting the requirements established at the start of the project.

Introduction

At the heart of the DGM system is the concept of data flow. Data flow programs differ from sequential programs in several ways; the first being that data flow programs execute when the data is available at any time during the mode whereas sequential programs have a predefined order of execution. Thus data flow exploits concurrency and parallelism and is suited for a multiprocessing environment. DGM pictorializes data flow in terms of graphs where arcs or queues represent the flow of data from software modules or nodes and control variables represent the exchange of information with the graph and the external world such as a radar operator or another graph. Figure 1 is a DGM graph of a typical radar search mode with algorithms such as the Fast Fourier Transform (FFT) and Constant False Alarm Rate (CFAR) represented as nodes in the graph. In addition, nodes may represent subgraphs, or an underlying directed graph, so that top down design may be utilized.

* Ada is a registered trademark of DoD.

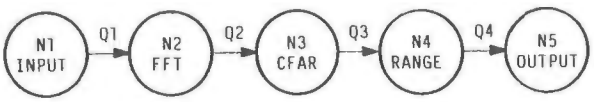


Figure 1. Directed Graph - SEARCH

The DGM Editor and Library Manager

The DGM Editor and Library Manager are interactive tools that enable the user to create a directed graph using data flow concepts and interface with the rest of the tools in the DGM system. In order to create a directed graph using the editor, definitions of software modules or, in particular, Ada packages or subgraphs, must be entered into the library. A library definition includes the unique name of the Ada package or subgraph and a description of the input, output and control parameters. Input and output parameters, or queues, are defined in terms of six attributes. These six attributes include the data type; threshold, or the minimum number of words that must be present before a node may execute; read, the number of words made known to the process at execution; consume, the number of words removed from memory at execution; produce, the number of words output from the executing node; and capacity, the amount of memory reserved for a queue, or the largest size a queue may become. Control parameter attributes include a name, data type, and mode. Mode may be in, signifying data is read-only, out, or write-only, and in out, or both readable and writeable from the node. A project may use multiple libraries, each interacting with the editor. One main library may contain all code that is under configuration control and is accessible for use by anyone, but available for change by only a few, while each individual may have his own library used for systems development and may contain packages not yet written.

Once a library of packages exist, a graph may be created. The first thing that must be defined in the editor is the library or libraries that will be used to extract package definitions. These definitions may not be changed while in the editor, thus ensuring that the original definitions for all packages will still exist once the graph is complete. Each queue is then defined in terms of data type, threshold, read, consume, produce, and two other

```
Summary of graph SEARCH

QUEUE THRESHOLD     READ CONSUME CAPACITY PRODUCE DATA-TYPE  INIT SOURCE      OSET SINK       OSET VALVE
Q1          128      128     128      356     128 SPFXC        F  N1          1 N2            1  ON
Q2         6272       49      49    18816     128 SPFXR        F  N2          1 N3            1  ON
Q3           49       49      49      147      49 SPFXR        F  N3          1 N4            1  ON
Q4           10       10      10       30      10 SPFXR        F  N4          1 N5            1  ON

NODE          STORAGE  PAC/SUB      1ST PROCESSOR  2ND PROCESSOR  EXCLUDES    SHARE PRIOR
N1                  0  INPUT        INPUT_BUFFER                              TRUE 0
N2               1424  FFT          CAVP           FPP                        TRUE 0
N3              37632  CFAR         CAVP           FPP                        TRUE 0
N4                294  *RANGE       FPP            CAVP                       TRUE 0
N5                 60  OUTPUT       OUTPUT_BUFFER                             TRUE 0

Total storage requirements for graph:      39410

End of graph SEARCH
```

Figure 2. Graph Attributes

attributes. The first additional queue attribute is initializable, indicating that a queue may contain data on it when the graph is instantiated, or first started. The second additional attribute that must be given is the state of the queue's valve. A valve that is off is analogous to a pipe turned off -- data will not flow through the queue. By default, all valves are on. Values for each attribute contained in the library may be taken as defaults or may be overwritten at any time.

Nodes are defined by attributes in the same manner as queues. The first attribute associated with a node is the name of the Ada package or subgraph that will be executing within the node. Once this name is given, the library definition for that package is extracted and input, output, and control parameters are prompted for based on the definition. For example, suppose a package in the library is defined as having one input queue of data type integer, and one output queue of data type complex. The user will be prompted for the name of a queue previously given that is of type integer to be used as input. He must enter a queue defined to be of type integer or the definition of the node will abort. A queue of the correct type may then be defined or another package may be used. Included in the definition of a node is the type of processor the node will execute in. Second, third, and fourth choices may be given indicating that if a processor of the first choice is unavailable, alternate choices are available for use. The exclude attribute indicates which, if any, nodes are excluded from executing within the same processor as the given node. Shareable, on the other hand, indicates whether any other nodes in the graph may share the processor with the given node.

Upon completion of the graph definition, the user may plot the graph. The editor initially places nodes on a grid and the user may keep this placement or rearrange any nodes. A file containing coordinates of starting and stopping points for queues and center point of nodes, diameters of nodes, and text such as the title of the graph, is produced. This file can then be used to produce a hard copy or terminal output such as represented in Figure 1. Additional files produced at completion include documentation listing all nodes and queues in the graph and their attributes as shown in Figure 2, Ada package specifications for every node in the graph, Figure 3, and files used to interface with several simulators. Automatically producing these files not only saves having to redefine the graph for each of the other tools, but saves the systems analyst from needing any knowledge of any other tool or a programming language. He can give the Ada specifications to several programmers to code the Ada body and the connectivity of the graph will be ensured.

Simulation

Because directed graphs are not sequential in nature and execution is therefore not easily traced, a variety of simulation tools exist to help analyze the correctness of the graph.

Data Flow Simulator

The DGM Data Flow Simulator provides the user with the facility to simulate the flow of data through the graph which has been defined using the editor and library manager. The data flow simulator passes data from queue to queue in the graph via the internal data structures of the simulator. During the simulation, the simulator keeps track of the number of times each node has fired (i.e. executed), the minimum number of data elements on each queue during a cycle, the maximum number of data elements on each queue during the cycle, and the current number of data elements on each queue in the graph. A cycle is the number of passes that go by between the display of output data to the screen. A pass corresponds to the simulator checking the node list to determine which nodes should be fired. The simulator requires a short driver program in order to run. This driver program is generated automatically from a question and answer session with the user. After the driver routine is generated, the simulator automatically compiles it and allows

```
with VECTOR_DEFS;

package N3 is

   task N3_TASK is

      entry GO_N3(

         -- input queues in package

         IN_QUEUE_1:  VEC_SPFXR_49;        -- Q2

         -- output queues in package

         OUT_QUEUE_1:  VEC_SPFXR_49);      -- Q3

   end N3_TASK;
```

Figure 3. Ada Package Specification

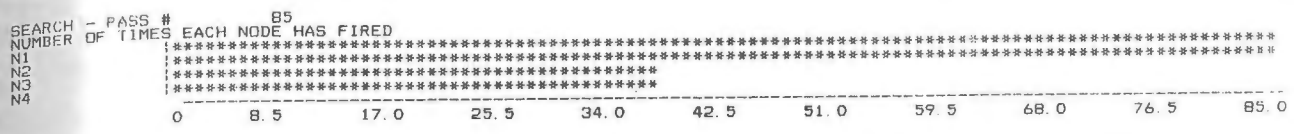


Figure 4. DGM Simulator - Histogram Output

the user to execute the simulation by making a selection from a menu.

The user may select from several options for displaying the results of the simulation. The displays fall into two categories; the first is a tabular format and the other a histogram plot. The data produced from the simulation is two sets of statistics, the number of data elements in each queue in the graph and the number of times each node has fired. The user receives the outputs once every cycle. The cycle length is specified by the user when he creates the driver program. For example, if the user specifies a cycle length of 5, output will be displayed after the 5th, 10th 15th, etc. passes of the simulation. When the user selects the display options that are desired, he may send the tabular and histogram data to files or to the console, as well as any messages telling him that the graph has deadlocked when such a situation occurs. Additionally, the user has the option of receiving the queue loading statistics in one of three forms: maximum during a cycle, minimum during a cycle, or current. The user may choose any combination of these outputs or all three to be displayed in the tabular and histogram formats. Examples of these two types of displays are shown in Figures 4 and 5.

Functional Code Simulator

The functional code simulator provides the user with the facility to emulate the functional execution of the graph. It makes calls to the actual Ada procedures which implement the functions of the nodes of the DGM graph. A package of Ada procedures implements the over eighty signal processing procedures executable in signal processing hardware. The simualtor keeps track of the data resident in the graph's queues while the graph is being simulated. Additionally, the simulator keeps track of the same queue loading and node firing statistics that were described previously with the data flow simulator.

The functional code simulator requires a driver program to execute the simulation in the same way that the data flow simulator does. This driver is also generated automatically by the simulator following a question and answer session with the user. After creating the driver program, the simulator

SEARCH - PASS # 85

QUEUE STATUS

QUEUE_ID	MINIMUM VALUES	MAXIMUM VALUES
Q1	0	0
Q2	128	9067
Q3	0	0
Q4	0	0

NODE STATUS

NODE_ID	FIRINGS
N1	85
N2	85
N3	37
N4	37
N5	37

Figure 5. DGM Simulator - Tabular Output

compiles the routine for the user so that the simulation is ready to be run.

The simulator also requires a procedure which makes the calls to the proper procedure when a node is ready to execute. This procedure is also generated automatically with the user needing only to supply the name of the graph summary file created from the editor (Figure 2). The simulator then automatically creates the procedure, places it in an Ada package and compiles the package for the user.

The user has several display options when using the functional code simulator. He may see the queue loading and node firing statistics which were available from the data flow simulator or he may examine data elements in any queue with his terminal or through a file.

Future Efforts

Future efforts in the area of DGM involve the implementation of a system design facility as shown in Figure 6. This design facility centers around the DGM Editor interacting with an expert system and several data bases. The expert system and each of the surrounding data bases and tools will be written in Prolog, an Artificial Intelligence language based on predicate calculus. A list of facts and rules comprising a Prolog data base is fed into the Prolog interpreter or compiler. Questions are then asked and based on the rules in the system, the interpreter either returns "yes" or "no" or supplies the missing information from the query; thus the given data base may be proven to be logically correct. As a beginning to this effort, the current DGM Editor produces a Prolog data base of the directed graph as in Figure 7. Facts, or attributes, of each node and queue are obtained from information supplied by the user when he is entering his graph into the Editor. This, then, is the "Graph Data Base" shown in Figure 6. The DGM Library, or data base of information concering packages and subgraphs currently being updated through the Library Manager, will become the "Nodal Data Base" of the expert system. A graphics package will enhance the DGM Editor so that the user may have a picture of the current state of his graph as he creates it.

The expert system begins by interacting with the systems analyst, the expert, through a natural language interface, enabling him to communicate with the program in a way he can understand. The knowledge of the systems analyst is absorbed into the "Expert Data Base" so that as the system grows, the expert system becomes able to create a graph for the user based on parameters supplied by him. For example, a user might ask for a graph representing a medium PRF radar search algorithm with 128 interpulse periods (IPPs), 49 range gates, and a data rate of 2 MHz. The expert system will then display a graph and ask if it is appropriate. The user may then add or delete functions and further refine the graph the system has chosen. The "Hardware Data Base" may then be used to determine the type of

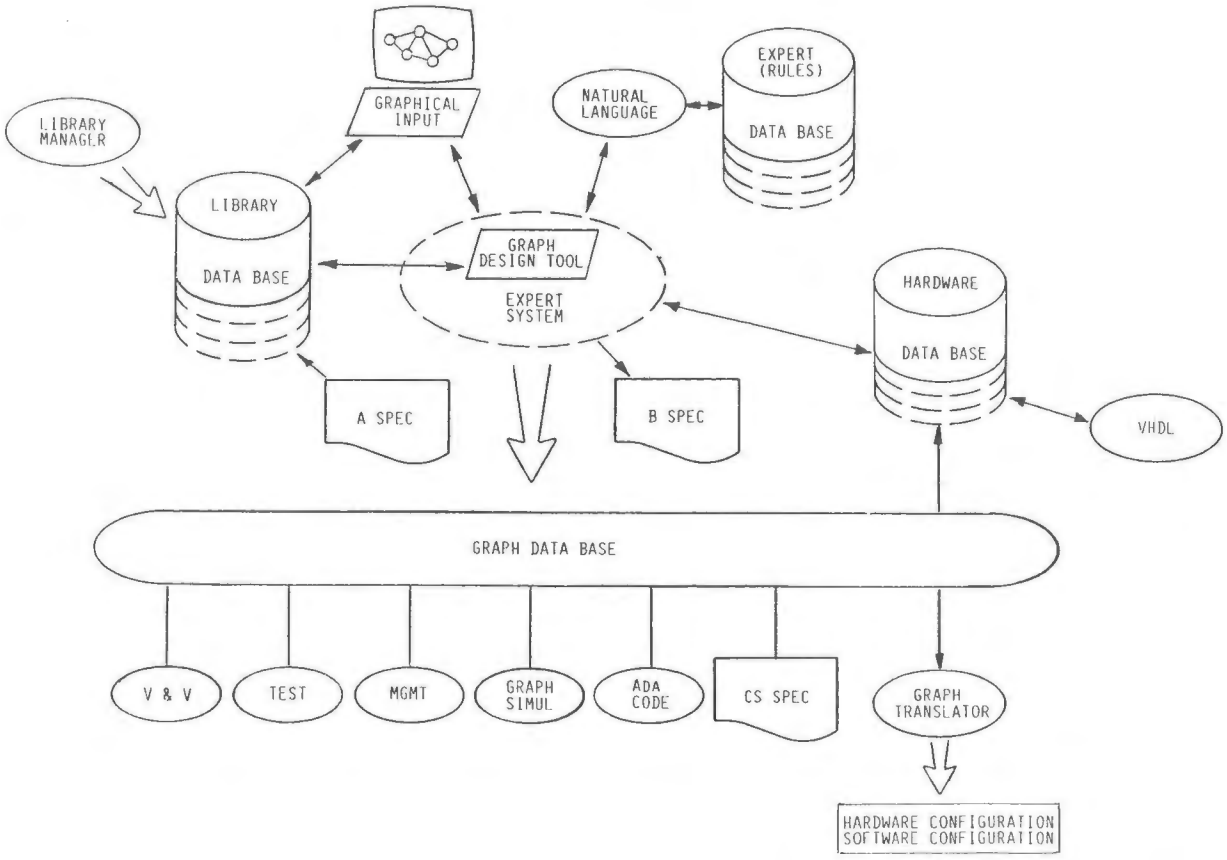


Figure 6. System Design Facility

hardware or hardware configuration necessary for the final graph. The Hardware Data Base would include information supplied by a hardware description language describing the capabilities of available hardware and would also include algorithmic information indicating what types of algorithms run efficiently on different hardware. A Graph Translator will use information in the Hardware Data Base and the Graph Data Base to partition the DGM graph across a hardware configuration based on processor utilization and will create the necessary data structures from this partitioning for the operating system.

Five additional tools written in Prolog will interface with the Graph Data Base. The first of these tools is an Ada code generator which will create compilable Ada code from a directed graph. Ada constructs such as decision structures will be represented through DGM constructs like valves whereas procedure calls and operations will be represented by nodes. The Ada program may then be verified and validated through another Prolog tool. The testing of software will also be managed through the expert systems environment. If a change is made in a directed graph, the portions of changed software will be highlighted as needing retesting. Along these lines, project management such as coding status or cost may be made visible through an additional Prolog tool. Graph simulators such as those mentioned above will also be implemented in Prolog.

Conclusion

Directed Graph Methodology is an integrated tool set that provides a means for system design in an efficient and descriptive manner. Using DGM ensures that each step in the design phase is validated and is consistent with previous and parallel efforts by providing automatically produced interface files that are changed correctly when a change is made to the original design. Future DGM systems carry this theory one step further by helping to design a correct system based on previous work, proven facts, and less human error.

```
/* Prolog database for graph SEARCH          */

/* A node in the graph is a fact */

/* node(<name>, attributes(...), [...input ports...], [...output ports...]). */

/* where attributes are */

/* attributes(<sub>,<pack>,<proc1>,<proc2>,<proc3>,<proc4>,<prior>,<excl>
<share>,<delay>)  */

/* and where an input port is  */

/* inport(<thresh>,<consume>,<capacity>,source(<node_name>,<offset>))  */

/* and where an output port is */

/* outport(<produce>,sink(<node_name>,<offset>))  */

graph(search          , [
n1            , n2            , n3            , n4
, n5            ]).

node(n1            , attributes(false, input       , '      *      ', '      *      ',
     input_buffer  , '      *      ', '      *      ',
                0, '      *      ', true,          0),
[],
[outport(       128, sink(n2            ,          1))
]).

node(n2            , attributes(false, fft         , '      *      ', '      *      ',
     cavp          , fpp           , '      *      ', '      *      ',
                0, '      *      ', true,          0),
[inport(        128,        128,        356, source(n1            ,          1))
],
[outport(       128, sink(n3            ,          1))
]).

node(n3            , attributes(false, cfar        , '      *      ', '      *      ',
     cavp          , fpp           , '      *      ', '      *      ',
                0, '      *      ', true,          0),
[inport(       6272,         49,      18816, source(n2            ,          1))
],
[outport(        49, sink(n4            ,          1))
]).

node(n4            , attributes(true, range        , '      *      ', '      *      ',
     fpp           , cavp          , '      *      ', '      *      ',
                0, '      *      ', true,          0),
[inport(        49,         49,        147, source(n3            ,          1))
],
[outport(        10, sink(n5            ,          1))
]).

node(n5            , attributes(false, output      , '      *      ', '      *      ',
     output_buffer , '      *      ', '      *      ',
                0, '      *      ', true,          0),
[inport(        10,         10,         30, source(n4            ,          1))
],
[]).
```

Figure 7. Prolog Database

84-2702

REAL LIFE CONSIDERATIONS OF ADA RUNTIME ORGANIZATIONS

FOR

REAL-TIME APPLICATIONS

J. Michael Kamrad II
Principle Research Scientist
Honeywell Systems and Research Center
Minneapolis, MN
(612) 378-4432
Kamrad at H1-Multics

Abstract

The Department of Defense has made it clear its intent to standardize all embedded computer system software in Ada. Developers of embedded computer systems for the DoD must now face the problem of how to make Ada meet their program requirements. To accomplish this developers will need to understand the importance that the runtime organization of our Ada software system has on the efficient execution of Ada software. A description of the Ada runtime organization is presented. The requirements for the runtime organization and the choices of implementations are explained and the importance of these issues to the success of meeting the needs of an embedded computer system are described.

What is Runtime Organization

The translation of Ada source programs into executable software requires two elements: a compiler and a runtime organization. The compiler performs the actual translation of the Ada program into an executable object program. The object program executes on the target computer environment defined by the runtime organization. No compiler, especailly an Ada compiler, generates object programs for a bare machine, because no known bare machine has all the functionality to support the Ada model of execution. As a result an Ada compiler generates object programs for an abstract "virtual machine" which is provided by the Ada runtime organization.

The Virtual Machine

The runtime organization provides all the facilities fo the virtual machine. These facilities consist of the instruction set architecture of the target computer plus a set of execution conventions and a set of runtime routines. The execution conventions interpret how the primitive resources of the target machine will be used. The set of runtime routines provide the missing functionality not found in the target machine. The effect of the runtime orgainzation is to add and delete facilities from the target machine to support the virtual machine for Ada program execution. As an example, execution convention may add data abstractions for arrays and records by defining how they would be represented, while runtime routines might provide the functionality to manipulate arrays and records. Conversely functionality may be eliminated as being undesireable or unneeded. Examples of this would be to eliminate the BCD instructions or privileged instructions from the objects programs generated from the compiler.

Ada Runtime Organization

The Ada runtime organization is distinguished from the runtime organizations of other language for two reasons. First Ada contains language features not normally found in other languages. Language features such as tasking, exception handling, and dynamic storage allocation demand additional functionality for managing processes, interrupts, and memory. These are features usually found in the executives of applications of general purpose operating systems. Second, Ada does not presuppose any existing operating system or executive. For an embedded computer system implemented in Ada this means the Ada programs must either depend on the existing executive to provide these functions or otherwise include these "executive" functions. If Ada runtime organization is designed to interface with existing real-time executives for embedded computer applications, it would expect the executive to provide the executive functions in a manner compatable with the language requirements. Unfortunately for all known real-time executives this compatability is poor at best, leading to intolerable performance.

As a result the runtime organization of Ada embedded computer appplication software contains these executive functions in a set of predefined runtime routines, often called the runtime kernel. This point is truly different and significant from past experiences when the selection and implementationof a program language was separate from design and implementation of the real-time executive. Now all these decisions must be made when the developer selects or directs the implementation of an Ada compilation system, in particular its runtime organization.

Impacts on Ada Runtime Organization

This section describes the impacts that the major features of Ada impose on its runtime organization. This explanation will make the reader appreciate the design decisions that must be considered in the design of the runtime organization and the consequence of unappropriate choices.

Tasking

The Ada Language Reference Manual defines a model of Ada execution. In this model a program is a collection of one or more tasks organized in a hierarchical tree structure. At the root of this tree is the "environment task" which conceptually surrounds an Ada program. An Ada program includes an environment of package libraries as well as the "main" subprogram. The activation of the "environment task" elaborates/creates the resources found in the Ada program context before it "calls" the Ada program for execution. The limbs of this tree represent tasks that are subordinate to the parent task at the root. The tree grows as new tasks are activated and shrinks as tasks are terminated. All the tasks execute asynchronously as if each task had its own dedicated processor.

There are five distinct operations associated with a task life cycle as Figure 2 indicates:

- Creation of a task
- Activation of a task
- Introduction of a task
- Rendezvous with task
- Termination of task

At the creation of a task, all the task entries are defined and an association with the task body is made. Additionally all the resources for the task are elaborated and the task's dependency on the parent task is made. The independent execution of each task is initiated at the end of the activation of the task, thereby establishing a execution life separate from its parent or siblings. The rendevous provides the means through which two tasks may communicate or synchronize with each other. Finally, at normal termination the task relinquishes its execution life and resources.

For efficiency reasons an interrupt may be treated in Ada as an entry call to a task. In this manner the software engineer may develop and represent interrupt handlers at the Ada source level rather than going outside the language.

Impacts. The Ada model for tasking makes two requirements on the runtime organization: mimic the "enviroment task" and provide the task operations. To mimic the "environment task" the runtime organization must perform the start and termination of an Ada program as the "environment task" would. At program start-up all the computer resources for execution must be acquired and all the program libraries are created (or "elaborated" in Ada nomenclature) before the main program is executed. Typically this requirement is handled by cooperation between the linker in the Ada compilation system and the Ada runtime routines. The linker will arrange or create object code that elaborates all the program libraries and the runtime kernel acquires all the necessary computer resources for program execution. In an embedded computer system this is trivial since by definition the application program has all the resources.

At either normal or abnormal termination all the resources used by the Ada program are "returned" to the computer system. Normal termination in Ada is when the "main" subprogram returns to the "environment" task while an abnormal termination is caused by an exception (an unusual problem) which has not been handled by the Ada program. For a dedicated computer system termination is accomplished by the runtime kernel halting execution of the computer. Yet embedded computer applications are never expected to terminate, even in the face of faults. In practice the runtime kernel must be prepared to perform, as the last resort, some fault tolerant action to either prevent termination or to terminate fault tolerately. If this fault tolerant response is application specific it must change with each application which, in turn, forces the runtime kernel to change.

All the tasking operations must be handled by the runtime kernel of the Ada runtime organization. Straight forward implementations of tasking operations in a runtime kernel have been shown to exist on all general purpose computers. Most of these are intolerably slow for embedded computer applications, especially as they support rendezvous between tasks. A number of optimizations have been proposed which work in specialized uses of tasks and rendezvous [HABE80, HILF 82]. It is here where considerable thought must be applied by Ada implementors to develop tasking operations with acceptable peformance. Judicious selection of the target computer is also very important since poor hardware support for tasking could make the implementor's job immensely more difficult. If the reader hasn't already guessed, this is a fruitful area for developing hardware to support context-switching, parameter passing and queuing that is required for tasking.

As an Ada entry call, an interrupt could be treated by the Ada runtime kernel like any other rendezvous. Unfortunately a pilot of an aircraft would appreciate more expedited response in the case of a threat-warning interrupt. Implementation of interrupt handling must provide that interrupt handling task be given control directly upon receipt of the interrupt, thereby, bypassing all the intervening queuing and dequeuing and scheduling that is required for other rendezvous.

```
task ADA_PROGRAM_EXECUTION;

task body ADA_PROGRAM_EXECUTION is
   package STANDARD is

      .
      .
      .
      <bodies for all library units including the >
      <   Main procedure                          >
   end STANDARD;

begin
   MAIN;
exception
   when others =>
      <handle all unhandled exceptions in a fault tolerant manner>
end ADA_PROGRAM_EXECUTION;
```

Figure 1. An Ada-like Description of Conceptual Anonymous Program Execution Task

```
program unit P is

   task T is                          --Creation of task
      entry X(K:ZERP);
      entry Y;
   end T;
      .
      .
      .
begin                                 --Activation of task T
   .
   .
   .
end P;
   .
   .
   .
begin                                 --Introduction of task
    accept X(K:ZERP)                  --Rendezvous
        do
        end X;
task body T is
      .
      .
      .
end T;                                --Termination of task
```

Figure 2. Life Cycle of Tasks

In the Ada model each task is assumed to have its own dedicated virtual processor. In actuality all these virtual processors are mapped by current implementation onto a single computer. Most embedded computer applications are implemented on multiple distributed computers. This requires a distributed Ada solution. An excellent paper by Dennis Cornhill [CORN 84] describes the most prominent approaches for partitioning an Ada software applications on ditributed targets. The first approach where there is one Ada program per target machine does not conform to the intended use of Ada and should only be used when the costs of support tools required by the other approaches can not be justified. The second approach where the Ada language is extended to include constructs for distribution allows the greatest freedom but certainly runs afoul of the DoD mandate against supersetting Ada. The third approach considers the application to be one Ada program and partitions the program along task boundaries. Although this approach has been the most frequently proposed it is cumbersome and not acceptable in most applications. The last approach appears to show the best promise. Unlike the third approach the fourth approach [CORN 83] can be used to partition an Ada program into any set of program unit or object that can be named. The user defines the partition in a separate specification which a specialized compiler would apply to the translation of the Ada program.

Memory

In the Ada model of program execution, memory is divided into two parts. One part contains storage for all data objects which are not dynamically allocated during program execution. Storage in this part is arranged in a tree structure that is homomorphic to the tree structure of tasks. All the data of the "main" subprogram, of the program libraries in the program context, and of the "environment" task are found in the root of the tree. Each new task activation adds a new limb to the tree to hold the task data and each subprogram call in that task extends the limb to include the data for the subprogram. The limb is reduced when the subprogram returns and is eliminated when the task is terminated. Visibility to data in the tree extends from any part in the tree up through the root.

Program objects which are dynamically allocated are obtained from the storage in the other part of memory, called a heap. More specifically the heap is an amorphous glob of storage which divided between collections of allocated objects with similar chararteristics and free storage. A collection is created when an access type which describes dynamically allocated objects is declared and is returned to free storage when the scope of the type is left. A dynamically allocated object is referenced through a pointer.

Impacts. The normal mechanism for implementing the storage tree is a "cactus" stack. The root stack contains all the same storage as the root of the storage tree. Each new task that is activated is given its own stack. Each stack is extended and reduced in the same classical manner that is used for all block structured languages. Storage for these stacks is obtained from unallocated computer memory and the heap is an excellent mechanism to supply this. Running out of free or stack storage is occurrance from which embedded computer applications must be protected.

Cactus stacks or equivalent approaches are difficult to implement on non-virtual memory machines, such as base register or extended memory machines, like the MIL-STD-1750A. This presents a dilemma because inefficiencies in storing and retrieving data extract a heavy burden on the system efficiency. The message here is to find computer whose memory scheme is better suited the memory model of Ada. Like tasking this problem is fertile ground for developing hardware to support this Ada memory model.

For validation of an Ada implementation a heap storage mechanism will be required. The Ada Language Reference Manuals(LRM) is notably vague about heap implementation except to say

- A collection exists until the end of the associated access type scope.
- A user may explicitly free (deallocate in Ada nomenclature) object in a collection.

The crucial concern is providing a limit to collection storage without running out of storage for any collection. Garbage collection is useful in identifying and releasing unreferenced collection objects for reallocation, but past appraoches are unacceptably slow for embedded computer applications. New software and hardware approaches to garbage collection which have been developed lately appear to provide better performance. Until these new approaches find their way into Ada implementation, Ada implementations should include these two optional language capabilities to help the user manage heap storage: explicit deallocation of access objects (Ada LRM-13.10.1) and limits on collection sizes (Ada LRM-13.2).

Exception Handling

An exception in Ada names a specific unusual or unexpected event. When the event occurs, an exception is raised. Raising an exception abandons normal program execution to draw attention to the event. Executing corrective action that responds to the exception is called an exception. Any program unit such as a subprogram, task or block may have an exception handler. When an exception is raised in a program unit with an exception ahndler for that exception, the handling of the exception terminates the execution of that program unit. Otherwise the exception simply abandons the execution of the program unit and propagates to the caller or to the program unit in which it is nested. If no exception handler exists for the exception in an Ada program or task, program or task execution is abandoned.

Impacts. Exception handling requires runtime kernel support. The dynamic nature of subprogram calling and task rendezvous prevents a solution that can be implemented completely in object code generated by a compiler.

Two conflicting approaches for implementing exception handling center around response time and execution overhead. One approach assumes that exceptions are rare. As a result there should be no overhead to be paid until the exception occurs. This approach will sacrifice response time when handling an exception to avoid the overhead. The second approach assumes response time is more important and so therefore will add any overhead that is necessary to meet its requirements. Developers for embedded computer applications must judge for themselves which approach is important. Support hardware could be developed which provide solutions that have good response time and, at the same time, no overhead.

Fault tolerance of Ada software is intergrally tied to the implementation and usage of exception handling. Because embedded computer applications can't terminate, the Ada program must have exception handlers to handle all faults; otherwise the runtime kernel has the responsibility of terminating program execution in a fault tolerant manner. As pointed out earlier, termination like this is application specific and is likely to cause changes in the runtime kernel.

Input/Output

In Ada input/output is not a part of the language. Instead all I/O is supplied in Ada packages. The Ada Language Reference Manual/(LRM) defines sequential, direct-access, text, and low level I/O as a set of predefined packages. Any other I/O facilities can be added as new Ada packages.

Impact. Although I/O is not part of the language technically, text I/O is required by the Ada Compiler Validation Capabaility test suite which is used to validate Ada implementations. This is an annoyance since most embedded computer applications have no need for text I/O or any of the other predefined I/O packages.

Developers of embedded computer applications may create additional I/O packages as their applications demand. Since implementation of these packages may be difficult to express in the Ada source language, the Ada LRM provides an optional features that allow access to either machine language or a foreign language (like FORTRAN, JORIAL, etc.) to express that implementation. Developers of embedded computer applications may be wise to insist that these features are included in their Ada implementations.

Otherwise is nothing technically challenging in I/O implementation, just a lot of work.

Conclusions

Standardization of Ada runtime organizations has been a subject that has been promoted in parts of the community. It is the author's opinion that standardization on one runtime organization is doomed to failure. Not every application wants or can afford the same performance or facilities from the Ada runtime organization. Instead what is anticipated are categories of Ada runtime organizations each of which is specifically suited to the needs of one application area such as C^3I, auronics, space, etc. The market place and user concensus is better suited in determining categories and nature of the Ada runtime organizations.

The reader can appreciated that the design of the runtime organization of an Ada implementation has great influence over the performance and therefore the success of the reader's application. A good, efficient implementation of tasking, memory management, etc. in Ada runtime organization cannot promise an successful application system but a bad implementation will surely guarantee failure. Therefore it is the responsiblity of each embedded computer system project to specify acceptable functionality and performance of the required and optional language features in their Ada runtime organization. Each project must assess the performance of current Ada implementations which may be acceptable but not optimal against the costs of development and revalidation that improvements to the runtime organizations of existing Ada improvements will require.

REFERENCES

[CORN 83] Dennis Cornhill, "A Survivable Distributed Computing System For Embedded Application Programs Written in Ada", Ada Letters, Volume III Number 3, November/December 1983.

[CORN 84] Dennis Cornhill, "Four Approaches to Partitioning Ada Programs for Execution on Distributed Targets", IEEE Computer Society 1984 Conference on Ada Applications and Environments.

[HABE 80] Habermann, A. N., and Nassi, I. R. "Efficient Implementation of Ada Tasks", CMU Technical Report CMU-CS-80_103, January 1980.

[HILF 82] Paul Hilfinger, "Implementation Strategies for Ada Tasking Idioms", Proceedings of the Ada TEC Conference on Ada, pp 26-30, October 1982.

USING ADA[1] FOR A DISTRIBUTED, FAULT TOLERANT SYSTEM

84-2703

J. Barton DeWolf[2]
Nancy M. Sodano
Roy S. Whittredge[3]

The Charles Stark Draper Laboratory, Inc.
Cambridge, Massachusetts 02139

Abstract

This paper presents a snapshot of one project's concerns and proposed solutions for using Ada to develop system software for a distributed, damage and fault tolerant processing architecture. The project is the NASA-sponsored Advanced Information Processing System (AIPS). The AIPS architecture is intended for advanced aeronautical and space vehicle applications. A proof-of-concept version of the system is being designed and built as a laboratory demonstration using MC 68010 processing elements. The system software manages system-wide, I/O network, intercomputer network, and local computer resources. Fault detection, identification, and recovery are provided through a combination of hardware and software features at each of these levels. This paper describes how the AIPS system software can be written in Ada using a standard uniprocessor runtime support package. The system software provides certain commonly-used services to the applications programmer beyond those inherent in the Ada language definition. In addition, it provides network transparency for interfunction communication. Implementing these services in Ada raises certain issues relating to the runtime support package. The current strategy for responding to these issues is described and recommendations are made for improving Ada runtime support features.

Introduction

Advanced avionics applications increasingly require underlying machine architectures that are damage and fault tolerant, and that provide access to distributed sensors, effectors, and high-throughput computational resources. In a recent survey of requirements for future space and aeronautical vehicle applications [1], allowable failure probabilities per mission ranged from 10^{-9} for a 10-hour mission with no repair (commercial transport aircraft), to 10^{-2} for a 20-year mission with repair (manned space platform). Throughput requirements ranged from 0.5 to 15 MIPS (million instructions per second), excluding specialized signal-processing requirements. The purpose of the NASA-sponsored Advanced Information Processing System (AIPS), currently in a laboratory model proof-of-concept phase at Draper Laboratory, is to provide a flexible, damage and fault tolerant architecture that can be tailored to meet the needs of such diverse applications.

After a detailed evaluation of six candidate languages, Ada was chosen as the language most suitable for implementing the AIPS system software [2]. The attributes of Ada that are most supportive of AIPS requirements are its provisions for (1) real-time programming, (2) error detection, handling, and containment, (3) modularity and separate compilation, and (4) standardization and portability. Its chief drawbacks at the present time are the limited availability and maturity of language implementations and support tools, and the limited experience in applying the language to real-time applications. This paper describes current plans for applying Ada constructs in the design of the system software for AIPS.

The next section provides an overview of AIPS, and the AIPS system software services and organization. Subsequent sections describe representative design issues in each of four major software categories. The reader is assumed to be familiar with the main features of the Ada language. The paper concludes with a summary of observations and conclusions.

Description of AIPS and the AIPS System Software

AIPS

The AIPS architecture consists of a distributed set of general purpose computers, an intercomputer network, one or more input/output networks, a mass memory, and system software to provide services for applications programs and to manage system resources. The architecture permits an application designer to select a set of hardware elements, establish an intercomputer (IC) network, establish input/output (I/O) networks, and to vary the redundancy level of these elements to meet application reliability requirements.

The proof-of-concept version of the system is shown in Figure 1. The five processing sites are comprised of a fault tolerant multiprocessor (FTMP), two triplex fault tolerant processors (FTPs), a duplex FTP, and a uniprocessor. The redundancy level at each GPC is allowed to vary to accomodate application functions having different reliability requirements. In addition, the redundant GPCs are designed so that the channels can be physically separated to provide damage tolerance. The sites are interconnected with a triplex circuit-switched nodal network. Input/output devices are connected to the GPCs using one or more simplex nodal networks. The AIPS mass memory employs coded redundancy and is connected to each of the GPCs using a triplex multiplex bus. The system hardware is described in greater detail in a separate paper in these proceedings [3]. To simplify the subsequent presentation, the operation of the FTMP and its system software is not considered.

The redundancy of the underlying hardware of the AIPS architecture is largely hidden from the system

[1] "Ada" is a registered trademark of the U.S. Department of Defense (AJPO).

[2] Member AIAA, IEEE.

[3] Member IEEE.

software and entirely hidden from the application software. All channels of a redundant GPC are synchronized at the instruction level and execute identical software. Fault detection and masking is implemented in the hardware through comparison of redundant results. A relatively small portion of the system software is devoted to fault isolation and reconfiguration (e.g., downgrading a triplex GPC to a duplex). Each channel of a redundant GPC transmits an identical data stream on a separate layer of the IC network. Each channel receives the data stream from all layers. Once again, fault detection and masking is mostly accomplished by the hardware. System software services are provided to diagnose and isolate faults in the IC and I/O networks, and to reconfigure the networks around faulty elements. These fault management activities are transparent to the application software.

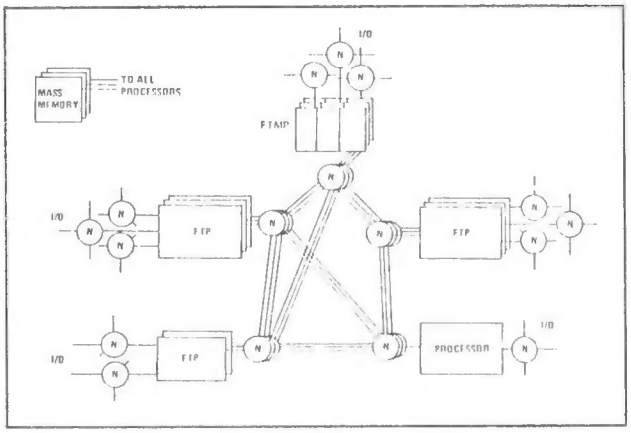


Figure 1. AIPS Proof-of-Concept Configuration

There is a requirement that the AIPS architecture not preclude the simultaneous use of multiple instruction set architectures for the different processing sites, to allow for the later insertion of improved technology. The proof-of-concept system will use MC 68010 processing elements for all the sites. For a similar reason, there is a requirement that the architecture not preclude the simultaneous use of multiple high order languages for the application software. Thus it cannot be assumed that future applications will be written only in Ada.

Services Provided

The AIPS system software provides numerous services in support of user applications. These services include functions invoked explicitly or implicitly by application software, as well as functions performed autonomously by the the system to maintain proper operation. These services have been organized into four broad categories as shown in Figure 2:

1. System Services
2. I/O Network Services
3. Intercomputer Network Services
4. Local Computer Services

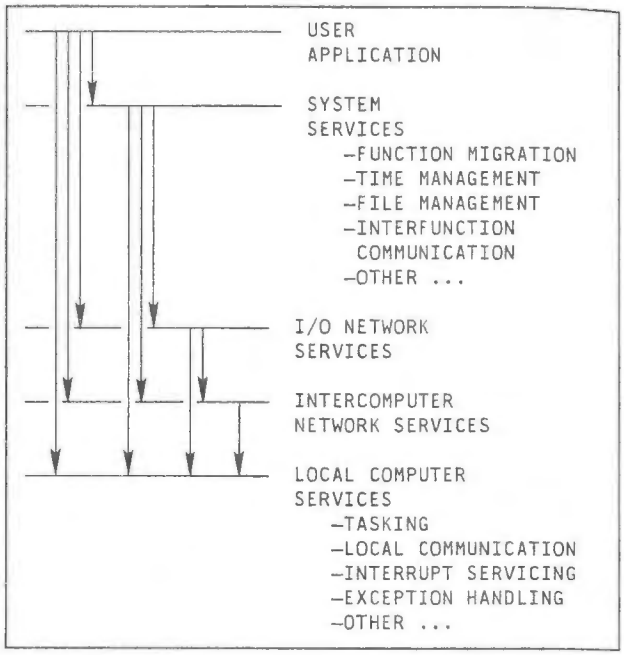


Figure 2. AIPS Services

To facilitate growth and change, testability, system integration, and the handling of software or other design faults, the system software has been structured as a series of layers, where each layer provides services to the layers above through well-defined interfaces. Since each layer hides implementation details from the other layers, the impact of a change which does not alter the interface is confined to the affected layer. Each layer is allowed to use only services provided by lower layers in the hierarchy. This means that if an interface is changed, it affects only the layers above.

The next sections of this paper examine the four layers of system services in more detail, and describe current plans for implementing the services using Ada.

Local Computer Services

In AIPS, many, if not all, of the required local computer services can be implemented using the currently defined Ada features [4] supported by a mature runtime support (RTS) package. In what follows, the requirements for selected local computer services are given along with representative implementations that satisfy these requirements.

Tasking

All AIPS tasks are scheduled and executed under the control of the local GPC. The requirements for task scheduling and dispatching are as follows:

- Tasks may be explicitly scheduled according to:
  - Priority
  - Time (one-shot and cyclic)
  - The occurrence of asynchronous prioritized events (one-shot and cyclic).

- Tasks may suspend themselves pending a time delay or event occurrence.
- A scheduled task (priority, time, or event) may be cancelled prior to task execution.

Although the Ada RTS package does not directly implement these tasking services, they may be easily provided in Ada using the services that the RTS does provide. Some representative implementations follow.

Priority Task. The general stucture of a task to be scheduled according to its priority is presented in Figure 3. Upon elaboration, the task PRIO_TASK will be activated and blocked at the selective wait construct "select accept PRIO_TASK_SCHED ... or terminate". Another task may explicitly schedule PRIO_TASK by performing a rendezvous with the entry PRIO_TASK_SCHED. If PRIO_TASK is of higher priority than the scheduling task, the scheduling task will be preempted and execution of PRIO_TASK will proceed. Otherwise PRIO_TASK will be placed on the RTS ready queue until it has the highest priority.

```
package PRIORITY_SCHEDULE is

  task PRIO_TASK is
    pragma PRIORITY(P);
    entry PRIO_TASK_SCHED;
    entry PRIO_TASK_CANCEL;
  end PRIO_TASK;

end PRIORITY_SCHEDULE;


package body PRIORITY_SCHEDULE is

  task body PRIO_TASK is

  begin
    loop
      select
        accept PRIO_TASK_SCHED;      --task scheduling entry
        select
          accept PRIO_TASK_CANCEL;  --task cancellation entry
        else

          -- task processing

        end select;
      or
        terminate;                   --task terminate point
      end select;
    end loop;
  end PRIO_TASK;

end PRIORITY_SCHEDULE;
```

Figure 3. Priority Task Structure

PRIO_TASK may be cancelled prior to execution by another task performing a rendezvous with the cancellation entry PRIO_TASK_CANCEL. At execution time PRIO_TASK will check for a rendezvous with PRIO_TASK_CANCEL using the select ... else construct. If no task is queued for a rendezvous, execution of the task proper will commence. Otherwise execution is not performed and the task is returned to the blocked state at PRIO_TASK_SCHED (until receipt of another scheduling rendezvous or a terminate condition).

Time Task. The general structure of a task to be scheduled according to time is presented in Figure 4. As with a priority task, upon elaboration the task TIME_TASK will be activated and blocked at the selective wait construct "select accept TIME_TASK_SCHED ... or terminate". To explicitly schedule TIME_TASK to execute periodically beginning at a specified time, a rendezvous is made with the entry TIME_TASK_SCHED with parameters

START_TIME absolute task initiation time.

PERIOD task execution cycle time (= 0 for one shot task).

```
package TIME_SCHEDULE is

  task TIME_TASK is
    pragma PRIORITY(P);
    entry TIME_TASK_SCHED(START_TIME: in TIME;
                          PERIOD: in DURATION);
    entry TIME_TASK_CANCEL;
  end TIME_TASK;

end TIME_SCHEDULE;

with CALENDAR;use CALENDAR;
package body TIME_SCHEDULE is

  task body TIME_TASK is
    NEXT_TIME : TIME;
    DELTA_T : DURATION;
  begin
    loop
      select
        accept TIME_TASK_SCHED(START_TIME: in TIME;      --scheduling
                PERIOD: in DURATION) do                  --entry
          NEXT_TIME := START_TIME;
          DELTA_T := PERIOD;
        end TIME_TASK_SCHED;
        while NEXT_TIME > CLOCK
          loop
            select
              accept TIME_TASK_CANCEL;                   --cancellation entry
              exit;
            or
              delay DURATION(NEXT_TIME - CLOCK); --time delay
              -- task processing
              NEXT_TIME := NEXT_TIME + DELTA_T;
            end select;
          end loop;
      or
        terminate;                                       --terminate
      end select;
    end loop;
  end TIME_TASK;

end TIME_SCHEDULE;
```

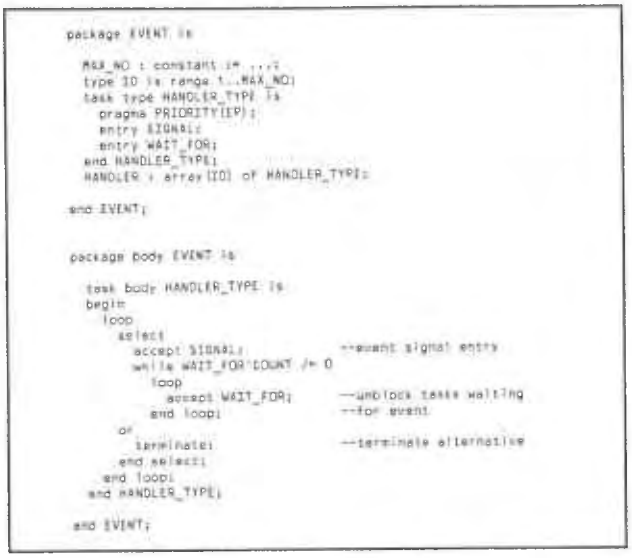
Figure 4. Time Task Structure

At the completion of the rendezvous processing, the task will be placed on the time queue via the delay statement. By expressing the delay time as the difference between the desired absolute start time and the current clock time, a correction is made for the overhead involved in the rendezvous.

Since the delay statement is an alternative to the cancellation entry TIME_TASK_CANCEL, a rendezvous with TIME_TASK_CANCEL any time TIME_TASK is on the time queue will result in TIME_TASK being removed from the queue and returned to the blocked state at TIME_TASK_SCHED (until receipt of another scheduling rendezvous or a terminate condition).

At the end of the delay interval, the execution of the task proper will begin. The task start time (NEXT_TIME) is updated at the end of task execution and, if a period has been supplied, the delay statement is executed again. The condition on the while loop ensures that a one shot task (DELTA_T = PERIOD = 0) will be executed only once.

Event Task. For a given application in AIPS, the number and type of events are predetermined. Therefore, it is possible to define a family of event handler tasks (one for each defined event) having the same relative priority structure as the set of events. The structure of the event handler task package is given in Figure 5. Each event handler task (EVENT.HANDLER(ID)) has two entries: SIGNAL and WAIT_FOR. At elaboration time, all handler tasks are activated and blocked at the SIGNAL entry.



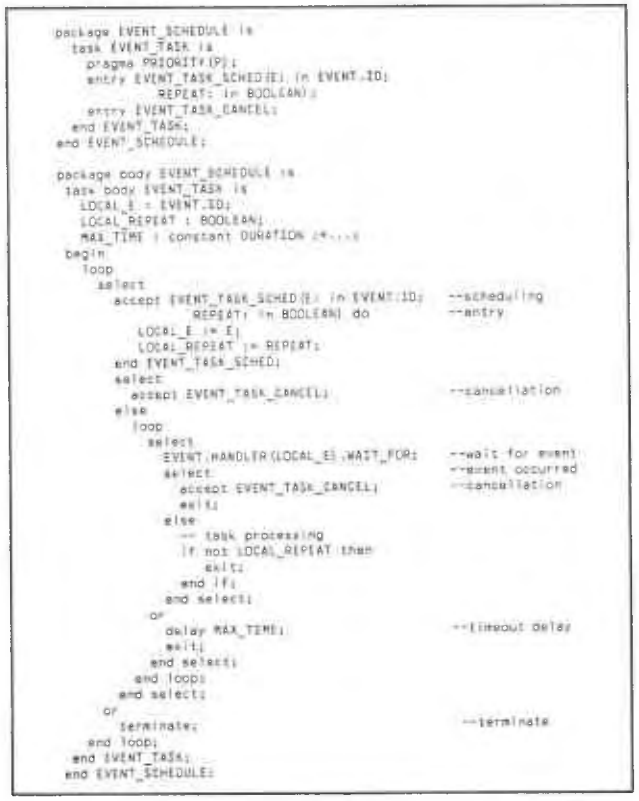
```
package EVENT is

  MAX_NO : constant := ...;
  type ID is range 1..MAX_NO;
  task type HANDLER_TYPE is
    pragma PRIORITY(P);
    entry SIGNAL;
    entry WAIT_FOR;
  end HANDLER_TYPE;
  HANDLER : array(ID) of HANDLER_TYPE;

end EVENT;


package body EVENT is

  task body HANDLER_TYPE is
  begin
    loop
      select
        accept SIGNAL;                  --event signal entry
        while WAIT_FOR'COUNT /= 0
          loop
            accept WAIT_FOR;            --unblock tasks waiting
          end loop;                     --for event
      or
        terminate;                      --terminate alternative
      end select;
    end loop;
  end HANDLER_TYPE;

end EVENT;
```

Figure 5. Event Handler Task Structure

The structure of a task to be scheduled pending the occurrence of a particular event is given in Figure 6. Again at elaboration time, EVENT_TASK will be activated and blocked at the scheduling entry EVENT_TASK_SCHED. EVENT_TASK may be explicitly scheduled by performing a rendezvous with EVENT_TASK_SCHED with the parameters

E event identifier.

REPEAT TRUE=cyclic event, FALSE=one shot event.



```
package EVENT_SCHEDULE is
  task EVENT_TASK is
    pragma PRIORITY(P);
    entry EVENT_TASK_SCHED(E: in EVENT.ID;
                           REPEAT: in BOOLEAN);
    entry EVENT_TASK_CANCEL;
  end EVENT_TASK;
end EVENT_SCHEDULE;

package body EVENT_SCHEDULE is
  task body EVENT_TASK is
    LOCAL_E : EVENT.ID;
    LOCAL_REPEAT : BOOLEAN;
    MAX_TIME : constant DURATION := ...;
  begin
    loop
      select
        accept EVENT_TASK_SCHED(E: in EVENT.ID;   --scheduling
                                REPEAT: in BOOLEAN) do   --entry
          LOCAL_E := E;
          LOCAL_REPEAT := REPEAT;
        end EVENT_TASK_SCHED;
        select
          accept EVENT_TASK_CANCEL;               --cancellation
        else
          loop
            select
              EVENT.HANDLER(LOCAL_E).WAIT_FOR;    --wait for event
              --event occurred
              select
                accept EVENT_TASK_CANCEL;         --cancellation
                exit;
              else
                -- task processing
                if not LOCAL_REPEAT then
                  exit;
                end if;
              end select;
            or
              delay MAX_TIME;                     --timeout delay
              exit;
            end select;
          end loop;
        end select;
      or
        terminate;                                --terminate
    end loop;
  end EVENT_TASK;
end EVENT_SCHEDULE;
```

Figure 6. Event Task Structure

After a scheduling rendezvous, EVENT_TASK will be blocked at the timed entry call (EVENT.HANDLER(LOCAL_E).WAIT_FOR) of the applicable event handler task. At any one time several event tasks may be queued at a particular event handler task WAIT_FOR entry .

When the event occurs, the event signalling task will unblock the associated event handler task by performing a rendezvous with the SIGNAL entry. The event handler task responds by unblocking all tasks queued at its WAIT_FOR entry in a first-in first-out manner using the COUNT attribute. If the priorities of the event handler tasks are sufficiently high, the event tasks will be placed on the RTS ready queue in order of priority. After unblocking all queued tasks, the event handler task returns to its blocked state at the SIGNAL entry point.

As in the case of a priority task, EVENT_TASK checks for a rendezvous with the cancellation entry EVENT_TASK_CANCEL prior to execution of the task proper. If a cancellation rendezvous is made, the main task processing is bypassed and the task is returned to the blocked state at EVENT_TASK_SCHED pending another scheduling rendezvous or terminate condition. If the repeat parameter (REPEAT) is true, EVENT_TASK will execute another WAIT_FOR entry call after task processing. This requeues the task for another event occurrence. Otherwise, the task returns to the blocked state at EVENT_TASK_SCHED.

Other Local Computer Services

Local Intertask Communication. The requirements for communication between tasks located on the same processor may be broken down into two general types: synchronous and asynchronous.

In synchronous communication, the communication event must be explicitly acknowledged by both sender and receiver tasks before processing is allowed to continue. In Ada, this is represented directly by the rendezvous construct. Each task will be suspended until both have executed the appropriate rendezvous statements: the entry call statement for the sender and the accept statement for the receiver.

Asynchronous communication involves interaction between sender and receiver via a shared mailbox. A sender task stores messages destined for a particular receiver task in the mailbox and continues execution without waiting for a message acknowledgement from the receiver. Similarly, the receiver retrieves messages from the mailbox without waiting for any explicit synchronization with the sender. Asynchronous communication is implemented in Ada via a rendezvous with a concurrent intermediary or agent task charged with managing the mailbox.

Interrupt Servicing. In Ada, interrupts are treated as calls to entries of tasks. The interrupt handler task is associated with the particular hardware address vectored to by the interrupt using the "for...use at..." construct. Execution of the task is blocked at an internal rendezvous point (accept statement). The occurrence of the interrupt results in a call being made to the entry of the handler task. The task may then accept the entry and service the interrupt.

Exception Handling. In Ada, an exception is defined as an event that suspends normal program execution. This may be an error or any other abnormal condition. In addition to the predefined exceptions (CONTRAINT_ERROR, NUMERIC_ERROR, TASKING_ERROR [4], etc.), the user may define his own using the exception declaration. The user-defined exception event is indicated by executing the raise statement. At the occurrence of the exception event, whether user-defined or predefined, execution is suspended and a user-supplied exception handler is executed. If an exception handler has not been defined at the program level at which the exception has occurred, the exception is propagated to successively higher levels of the program structure until an appropriate handler is encountered.

Intercomputer Network Services

Requirements. The intercomputer (IC) network services provide for reliable communication across processor boundaries, and may be divided into the services that provide and maintain the reliable medium for communication, and the services that perform the communication itself.

Design. The reliability services will manage the network of nodes linking the processors, so that a path is available, as needed, for communication between application software functions residing on different processors. The failure of a node will not eliminate access to a processor, when an alternate path to that processor can be made available. The reliability services will be invisible to the application software.

The communication services will be based on the layering concept described in the International Standards Organization Open Systems Interconnection Basic Reference Model (ISO OSI) [5]. The communication services will provide the data transmission services to support the function migration, time management, file management, and interfunction communication services (discussed below). They will not, in general, be accessible to the application software directly.

Implementation. Apart from the issues discussed below for the implementations of the system services, no difficulties are envisioned in implementing in Ada the IC network services themselves.

I/O Network Services

Requirements. These services provide for all communication between application software and I/O devices, over a reliable network.

Design. The reliability services are similar to those of the IC network services. The communication services, like the IC network communication services, will be based on the ISO OSI layering concept. Unlike the IC network communication services, the I/O network communication services will be directly accessible to the application software.

Implementation. The reliability services can be implemented in Ada, though they require use of the communication services.

One way of implementing the communication services is by modifying the parts of the uniprocessor runtime support packages that implement the Ada predefined I/O packages SEQUENTIAL_IO, DIRECT_IO, TEXT_IO, IO_EXCEPTIONS, and LOW_LEVEL_IO [4] to support the network communication methods. This approach can be implemented using the subset of Ada which does not include the predefined I/O packages, to write protocol handlers, device drivers, etc. It would require using an Ada system with a modular runtime support package with well-defined interfaces, and a means for incorporating Ada modules as part of the runtime support package. This implementation alternative would make the communication services completely transparent to the application software, by providing access through the predefined interface.

Another way of implementing the communication services is by providing a high-level Ada package interface between the application software and the I/O devices. This would require the application software to use the package defining these services instead of the communication handling parts of the predefined I/O packages; and it would require the services to provide support for all the communication modes defined in those packages. Presumably, the body of the communication services package could then be written entirely in Ada, bypassing the predefined I/O packages and interfacing directly to the network via representation specifications. This implementation alternative places restrictions on the application software which would have to be checked, and may require some modifications to the runtime support package as well. Note that it is still necessary to generate the system to support the particular I/O configuration for the application, by writing device drivers, including them in the system load, and making them known to the areas of the operating system which use them.

System Services

Function Migration

Requirements. This service provides one of the basic means for tolerating hardware failures in the system. The application software will be divided into conceptually unified areas, called functions, such as Guidance or Navigation. If the reliability of the current system configuration falls below that required for an application software function, the system software may reconfigure the system to meet the requirements of the function. The system may fail to meet the reliability requirements, for example, due to degradation of the particular processing site upon which the function is executing, or degradation of one of the communication networks which it uses.

System reconfiguration may include changing the logical connections between hardware elements in the system, and "moving" application software functions from one processing site to another with sufficient reliability and access to the communication networks. This moving of application software functions is provided by the function migration service. The function migration service may also be used in response to mission events or time, or at operator request.

Design. The sites on which a particular application software function may execute will be determined in advance, and the code for those sites will be prestored in the system. Thus, in its simplest form, "moving" a function comprises letting it reach a quiescent point in its execution, saving any data which will be needed for the new embodiment of the function, deactivating the copy on the original processing site, restoring the saved data to the copy on the new site, and activating the copy on the new site. (In addition, any system data areas which contain addressing information for the function will be updated to reflect the new placement. See "Interfunction Communication", below.)

The function migration service does not have a visible interface with the application software functions when it is used to mask failures. The application software may invoke the service through an interface which will be provided, when it is used to respond to the occurrence of a mission event or time. The configuration which the application software requests, however, must be one of those specified when the application system was designed.

Ideally, the failure support part of this service would be completely invisible to the application software writers, and would allow code sections smaller than functions to be moved. This would require solutions to several unwieldy problems. First, each relocatable unit would have to be stored in the form necessary for each machine. Then, each relocatable unit would need to be analyzed for state information, which would need to be saved and restored if the unit were moved. Finally, having the system decide which units to combine in a given machine at a particular time is a difficult linear programming problem. Relocating at the function level is believed to be a reasonable compromise in functionality for space and operating system complexity.

Implementation. Given the support of the interfunction communication, intercomputer network, and local computer services, no impediments to implementing this service in Ada have been identified at the current level of design. It is possible that the service will need to make extensive use of machine dependent features, through representation specifications, to handle halting processors, placing code in precise areas of memory, and manipulating the processor state. It is also possible that the service will need to know the location, format, and content of the Ada system state, to be able to stop and start moved functions quickly enough, without disrupting the rest of the system. The alternative is to require each application function to provide means for saving its state in an external area, halting itself, restoring its state, and restarting itself.

Time Management

Requirements. This service provides a common time to all the processors, with granularity and skew as specified in the performance requirements. The system software treats the system time as an entity logically independent of local time, provides a means for accessing the system time, and a means for keeping the local time within a given tolerance of the system time.

Design. In AIPS, one time source will be designated the current system clock. Its value will be broadcast periodically to all processors in the system, which will use that value to update their own clocks. The system software will not provide the application software with an interface to system time, as distinct from local time: the local time will be updated automatically by the system software to reflect the system time.

Implementation. The Ada semantics provide access to clock characteristics through the hardware-dependent subset of the Ada constructs dealing with time: the function CALENDAR.CLOCK, the type STANDARD.DURATION, and the named number SYSTEM.TICK [4]. The Ada semantics do not provide the distinction between system and local time, so for a system with hardware which makes the distinction meaningful, the constructs listed above may be implemented to refer either to the system time or to the local time. An implementation which referred to the local time without providing the additional capability of updating the local time from the system time would be insufficient for a large class of real-time applications, where coordination across processors is required. Given periodic updating to the system time, local clocks may be used to support CALENDAR.CLOCK. Their characteristics would then be available through STANDARD.DURATION and SYSTEM.TICK. This would allow rapid access to the time, but reduce the portability, within the system, of code which depended on the DURATION range and TICK values.

At this stage of development, the hardware and software designs for this service are not complete, so it is not clear whether or not it can be implemented in Ada. The interface it presents to users of the system can certainly be specified in Ada. One area of concern is the interface between the code implementing the service, and the compiler-runtime support package. The defined Ada semantics do not provide for write access to the clock. It is possible that this interface may be defined with representation specifications, but cooperation is required between this service and the runtime support package to maintain the interface, and to ensure that the desired effect is achieved by updating the clock.

File Management

Requirements and Design. One of the design goals of the AIPS system is to provide to the application software the interface of the Ada semantics as defined in the Language Reference Manual [4]. This service will supply the defined Ada file interface, where a file might reside in another processor, mass memory, or be read or written to an I/O device.

Implementation. Similarly to the I/O network services discussed above, the file management services can be implemented in two ways: by providing a high-level Ada package interface between the application software and the interfunction communication, I/O network, and IC network services; or by modifying the parts of the uniprocessor runtime support packages that implement the file handling services of the Ada predefined I/O packages to refer to the interfunction communication, I/O network, and IC network services.

The discussion given for the I/O communication services is equally valid here: The body of the file management service package can be written in Ada, by bypassing the predefined I/O packages and interfacing directly to the Ada interfaces of the lower level system services. Modifications to the runtime support system can be coded in Ada to the extent that that code can be integrated with the RTS.

Interfunction Communication

Requirements. This service directs all communication between application software functions. It keeps track of the current location of each application software function, and resolves references from one function to another to the correct processing sites. It uses lower level communication services to perform the actual data transfers; either the local computer communication service for communication within the single processor, or the IC network communication service for communication across processor boundaries.

Design. The interfunction communication service can be viewed as the major piece necessary to complete the Ada runtime support package for a distributed system, by providing the interface between the application software functions and the currently available uniprocessor runtime support packages. If implemented in full generality, the service would include support for all the Ada mechanisms for communicating across program unit boundaries, such as object creation, deletion and access; subprogram invocation; parameter passing; task activation and abortion; entry calls; file access; and exception propagation. Discussion is currently underway about the necessity and practicality of supporting this full set for AIPS.

Implementation. There are three methods under consideration for implementing the interfunction communication service for AIPS:

(1) Similarly to the file management service, one method is through modifying the uniprocessor runtime support package so that every reference to an entity across a function boundary is funnelled through an address resolution mechanism.

This has the desirable effect of being completely transparent to the application software, except with regard to timing behavior, and does not perturb the development process. It is not clear, however, that the implementation of the service can be limited to the runtime support package interfaces to the application software. If, for example, the compiler assumes all procedures in a package are to reside on the same machine, and generates absolute addressing within the package, moving a procedure to another machine would make control over the code generation process necessary. This might be quite expensive; and would impose a higher maintenance cost when compilers are upgraded or a new machine type is introduced into the system.

There do not seem to be any difficulties in building the address resolution mechanism, itself, in Ada. Connecting it to the runtime support package, and modifying the compiler's code generation capability have not been investigated in detail.

(2) The other extreme of implementation methods is to provide an Ada package interface which an application function must use when communicating with another function. The body of the package could be written in Ada, to refer to the IC network service interface.

This method is the easiest to implement, and requires least interference with the runtime support package, but it forces the application programmer to know where the function boundaries are, and explicitly use different methods for what he or she might normally think of as standard Ada constructs. Also, adherence to this convention might be difficult or expensive to check.

(3) A middle-ground approach [6] is to allow the programmer to write as if the first implementation were in effect, and then preprocess the source code to turn it into what would have been written had the second implementation been in effect. This provides the clean user interface, and allows an implementation which does not interfere with the code generation or runtime support package, but it does require that the preprocessor be able to recognize constructs which require cross-function communication.

Summary and Conclusions

In this paper, we have given a capsule description of the AIPS system, described the system software services which will be provided for AIPS, and discussed implementation alternatives and issues involved in using Ada to develop the system software. Table 1 summarizes the ways in which the AIPS system software services may be implemented. In the table, implementation alternatives which are discussed in the text are shown as numbered alternatives under the service headings. The first column ("Part of U-RTS") is checked if the service is provided as part of the currently available runtime support packages for uniprocessor Ada systems. The second column ("Ada & U-RTS") is checked if the service can be implemented in Ada without recourse to knowledge about the structure or content of the RTS. The third column ("Modif'd U-RTS") is checked if knowledge of the RTS is required, or if the service presents a standard Ada capability which can be implemented by providing support in the RTS for the distributed architecture.

The local computer services for local communication, interrupt handling, and exception handling are part of the Ada language definition, and are all provided as part of the currently available runtime support packages for uniprocessor Ada systems. The actual interrupt and exception handlers are not provided, and can be written in Ada.

The local computer tasking services can be written in Ada if a suitable implementation of the priority pragma is provided, and the rendezvous implementation meets performance requirements.

The intercomputer network reliability and communication services, and the I/O network reliability services can all be written in Ada, given an implementation that provides representation specifications which can be used for I/O. The system function migration services, as they are understood today, can also be written in Ada. Details involving controlling the states of the processors remain to be

resolved. They may require interaction with the RTS, as indicated by the "?" in the table.

The I/O network communication services and system file management services are closely related. Together they are analogous to the services provided in the Ada predefined I/O packages. If they are implemented as providing the support for the predefined packages, then modifying the uniprocessor RTS to support the distributed system may be feasible. This may be done in Ada if the Ada system allows inclusion of Ada code in the RTS. If these services are implemented as a higher level utility package, then the package itself can be written in Ada, but may require some supporting work in the RTS. The predefined package implementation for file management is preferred because the Ada file interface closely fits with its definition. It is not yet clear that this is the case with the I/O network communication service, so neither implementation is preferred.

The interfunction communication services present a similar case to the file management and I/O network services. They can be implemented in Ada without reference to the RTS, but they provide an extension, for the distributed hardware architecture, of the Ada means of communicating across module boundaries. If implemented as part of the RTS, they may be completely transparent to the user.

The time management services require modification to the RTS, since the Ada language definition does not provide write access to the clock.

Table 1. Implementation Options for AIPS System Software Services

	Part of U-RTS	Ada & U-RTS	Modif'd U-RTS
Local Computer Services			
Tasking		X	
Local Communication	X	X	
Interrupt Servicing	X	(handler)	
Exception Handling	X	(handler)	
Intercomputer Network Services			
Reliability		X	
Communication		X	
I/O Network Services			
Reliability		X	
Communication			
1. Predefined packages			X
2. Higher-level interface		X	
System Services			
Function Migration		X	?
Time Management			X
File Management			
1. Predefined packages			*
2. Higher-level interface		X	
Interfunction Comm.		X	*

Key:
U-RTS - uniprocessor runtime support package
X - possible
* - preferred

In conclusion, Ada appears to be suitable for implementing the services which will be provided by the AIPS system software, with the following caveats:

- Currently available compilers and runtime support systems do not support distributed or fault tolerant hardware configurations.
- Runtime support systems should support the defined Ada language features in a modular fashion, with well-defined and documented interfaces.
- Environments for developing real-time embedded systems with Ada should include means and tools for incorporating application-specific modules into the runtime support package.

Acknowledgement

This report was prepared by The Charles Stark Draper Laboratory, Inc. under Contract NAS9-16023, Task Order #84-18, with the Lyndon B. Johnson Space Center of the National Aeronautics & Space Administration. Publication of this report does not constitute approval by the NASA/JSC of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

References

1. AIPS System Requirements, AIPS-83-50, 30 August 1983, The Charles Stark Draper Laboratory, Inc., Cambridge, MA.

2. A. A. Knosp, Jr., AIPS Language Trade Study, CSDL-C-5694, March 1984, The Charles Stark Draper Laboratory, Inc., Cambridge, MA.

3. J. Lala, "Advanced Information Processing System", Proc. 6th AIAA/IEEE Digital Avionics Systems Conference, December 1984, Baltimore, MD.

4. Ada Programming Language, ANSI/MIL-STD-1815A, January 1983, American National Standards Institute, Inc.

5. Information Processing Systems - Open Systems Interconnection Basic Reference Model, Draft International Standard ISO/DIS 7498, April 1982, International Standards Organization/Technical Committee 97.

6. D. S. Lane, G. Huling, B. M. Bardin, "Implementation of A Real-Time Distributed Computer System in Ada", Proc. 4th Computers in Aerospace Conference, October 1983, Hartford, CT, Collection of Technical Papers (A84-10001 01-59), AIAA, New York, 1983.

SESSION 17 ON-BOARD MONITORING AND TEST

Chairmen:

L.M. Carrier
Rockwell International Corp.

D. Pieratt
ASD/B1EE

17

This session presents current and future approaches for implementation of on-board monitoring and test systems, including technical and operational issues of fault monitoring, isolation and system maintenance.

MAINTENANCE ASSIST FUNCTIONS EMBEDDED WITHIN THE 737-300 FLIGHT MANAGEMENT SYSTEM

Gordon F. Ellis
Specialist Engineer - The Boeing Company
Seattle, Washington

Henry E. Hofferber
Staff Engineer - Sperry Flight Systems
Phoenix, Arizona

Abstract

The paper presents the design, and design philosophy, of a "federated" maintenance system employed on the Flight Management System (FMS) on the new Boeing 737-300 commercial jet transport. Covered are the design guidelines followed in developing the system, and how the airline customers are envisioned to utilize the system in practice. The design of the Maintenance Assist Function is then discussed for the more complex of the "federated" built-in test (BITE) systems, namely the Digital Flight Control System (DFCS). Within the discussion, the design for Ground Maintenance BITE, in-flight Continuous Monitoring, and the Maintenance Monitoring fault storage and readout is detailed. Finally, the paper addresses the experience gained by the development team, with the introduction of sophisticated BITE systems, in the new airplanes that are equipped with digital avionics. In general, the paper provides a baseline "federated" BITE concept as applied to the major onboard computing systems of the FMS. However, a conceptual foundation is provided which could be expanded to cover more onboard avionics systems as future commercial airplane types are developed.

Introduction

The Boeing 737-300 Flight Management System (FMS) includes a sophisticated onboard BITE system embedded within each of the major computer systems comprising the Flight Management System. The BITE provides an integrated ground maintenance/in-flight maintenance monitoring system that is available to the maintenance personnel whenever power is applied to the aircraft. The system design objectives are to minimize on-airplane maintenance time, reduce unconfirmed line replaceable unit (LRU) removal rates and facilitate identification of failed LRUs and associated interfaces within the Flight Management System.

Federated BITE Concept

All of the major FMS computer units — Digital Flight Control System (DFCS) flight control computers (FCC-A and B), Auto Throttle Computer (A/T C), Flight Management Computer (FMC), and Digital Analog Adapters (DAA 1 and 2) computers — have extensive BITE programs. (Note that the IRS BITE is contained within the DAAs.) These BITE systems are controlled and displayed at a single location within the cockpit. Therefore, integration of the four federated BITE systems is achieved at the time-shared FMC-CDU (control display unit). (See Figure 1.) The line maintenance technician would access any of the four BITE systems from the centralized location. Selection of an individual FMS BITE (DFCS, A/T, IRS) causes a BITE turn-on code to be transmitted to the respective computer units. If the necessary safety preconditions are satisfied, the selected computer directly communicates ARINC 429 ISO ALPHABET No. 5 messages to be displayed on the FMC-CDU. FMC BITE selection results in internal BITE activation, with the associated FMC-BITE menus being displayed. Having a federated BITE concept allows for each subsystem to tailor its BITE to the specific needs and idiosyncracies of that system. Each of the four "federated" BITE systems has been designed to provide common functions at the Line maintenance/Overnight maintenance levels, which leads to consistency of operation for the technician. However, due to different system peculiarities, departure from a common format is desirable at the Engineering/Specialist Technician level. This paper will now address the DFCS BITE, on how the Ground Maintenance BITE and in-flight Maintenance Monitoring was configured and designed. The maintenance assist functions of the other systems are similar but are not as extensive or of the same complexity as the DFCS.

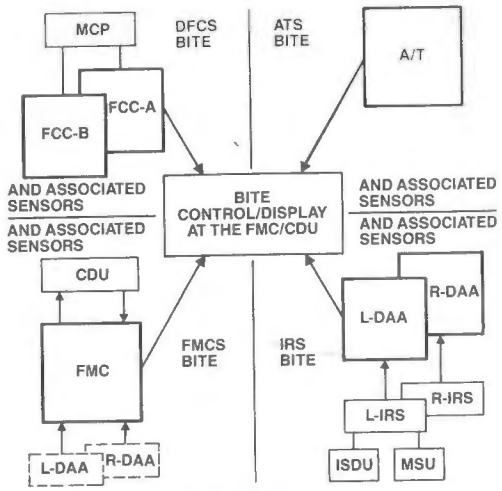


Figure 1. Federated Bite Concept

DFCS BITE Design Guidelines

The DFCS BITE is designed to be consistent with the following overall guidelines.

- BITE shall be consistent with the "ON CONDITION" maintenance philosophy. BITE activity occurs after a malfunction or squawk has been reported via a Pilot Report (PIREP).
- Capability of detecting and isolating 95% of all internal steady state failures to the LRU level. (As an objective, isolate 95% of steady-state system failures.)
- Rapid fault isolation on the ground in response to a given PIREP. (Objective is to isolate a single failure to an LRU in less than 3 minutes.)

- Designed for simple operation by a line mechanic with minimum avionics training. All messages to be in plain "English language" with no codes displayed to the maintenance technician.
- All in-flight faults are to be correlated with a cockpit effect.
- No carry-on test procedures required at the line maintenance level. Documentation available for interpretation only at the specialist or engineering level.
- One-man operation from the Flight Deck. Two-man operation shall only be required for safety reasons when surface motion tests are performed.
- No ground support test equipment required (except in the case of the sensor/surface rigging tests).
- No flight crew-initiated tests are required.
- Provision of postmaintenance verification tests to check interfaces and for autoland reinstatement.
- Ability to conduct a complete ground functional test of the entire system, i.e. all systems that are provided within the DFCS Flight Control Computers.
- Provide rigging checks of the dedicated surface sensors/actuation systems.
- In-flight fault recording into nonvolatile memory utilizing on-line continuous monitoring or dedicated monitoring specially provided for the Maintenance Monitoring task.
- Diagnostics are to indicate (in plain English language) which LRUs are at fault down to a single LRU to the maximum extent. FCC diagnostics shall be included if internal circuitry can contribute greater than a negligible probability.
- All hydraulic actuation tests shall be grouped together to minimize hydraulic power application time.
- BITE shall be designed for different levels of operation. These levels would be for line maintenance personnel during quick and extended stopovers and for engineering investigative purposes.
- Ground BITE shall be positively deactivated during the Autopilot/Flight Director operational envelope.
- Utilize the basic guidelines and recommendations outlined in ARINC 423 and ARINC 701 characteristics.

DFCS BITE Design

Maintenance Cycle

The maintenance cycle begins with a verified system working satisfactorily (see Figure 2). This next phase is fault identification where the flight crew reports a malfunction that initiates maintenance action on the ground. For the "on ground" fault isolation phase, BITE requires a means to review the IN-FLIGHT maintenance monitor fault storedowns, and/or a rapid fault isolation test sequence. After removing and replacing the failed LRU, the system interfaces and/or autoland status must be reverified by an "on ground" procedure. The malfunction has now been cleared and the system verified to resume flight operations.

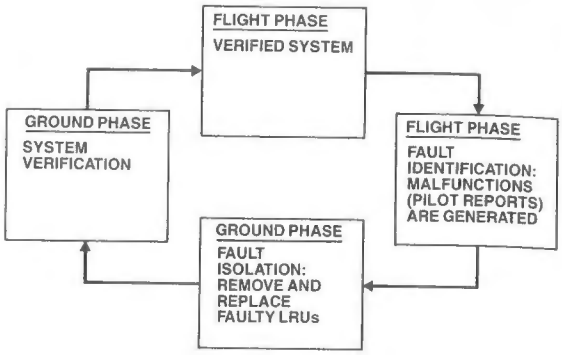


Figure 2. Maintenance Cycle

Maintenance Assist Functions Within DFCS

To provide maintenance assist functions with the DFCS to cover all the requirements imposed, three major areas of BITE/BIT design were provided. These are: 1) Ground Maintenance BITE, 2) Continuous Monitoring including periodic checks at POWER-UP, 3) Maintenance Monitoring. As shown in Figure 3, these functions are intertwined in actual practice. Maintenance monitoring utilizes the basic health monitoring of the Continuous/Power-Up Monitors. Ground Maintenance BITE is used to access the in-flight fault results and exercises the Continuous/Power-Up test routines as part of the Ground BITE-activated isolation tests.

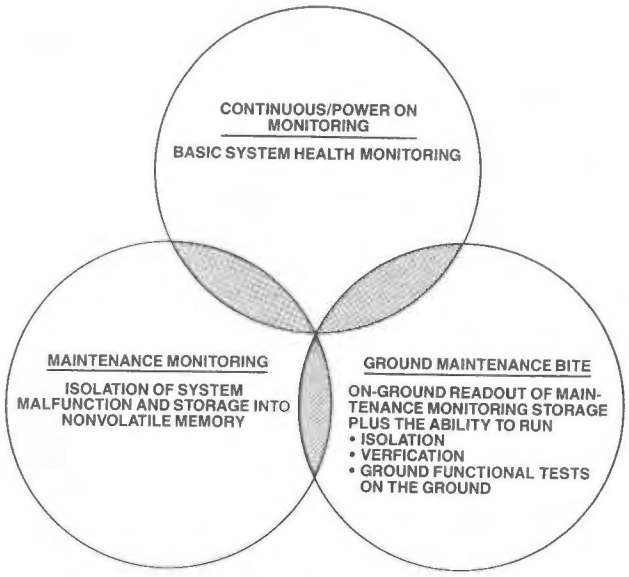


Figure 3. DFCS Maintenance Assist Features

BITE Levels of Capability

BITE was designed for three operational levels to satisfy the needs of the line maintenance technician at the line station, or for extensive checkout where time of completion purposes is not as demanding. Investigative capability by specialist technicians/engineers into intermittent type malfunctions or faults that are difficult to isolate is also provided.

Level 1 Line Maintenance BITE

Figure 4 depicts the LINE MAINTENANCE features which are used to rapidly isolate the cause of a fault and provide postmaintenance verification during short aircraft turnaround.

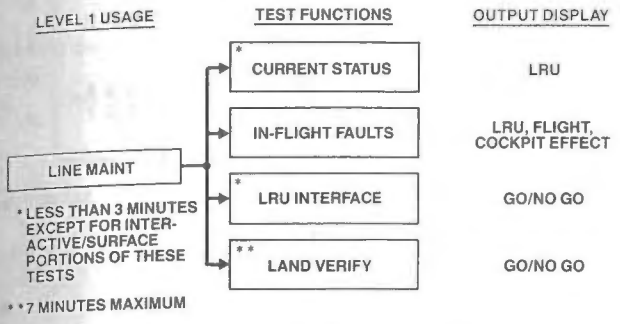


Figure 4. Line Maintenance Bite

Level 2 Overnight Maintenance

Overnight maintenance capability exists by using the functions provided in Figure 5. These are comprehensive tests for longer maintenance periods.

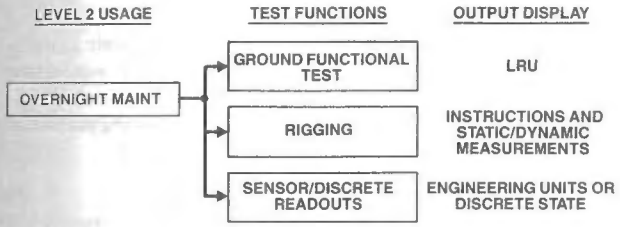


Figure 5. Overnight Maintenance

Level 3 Engineering Investigative Capability

This level provides for readout of individual test results with displayed nominal values and/or upper and lower limits. Memory interrogation is provided to access the contents of ROM, RAM, and nonvolatile EEPROM, and also to display the in-flight fault history and monitor that tripped during in-flight fault storage. (See Figure 6.)

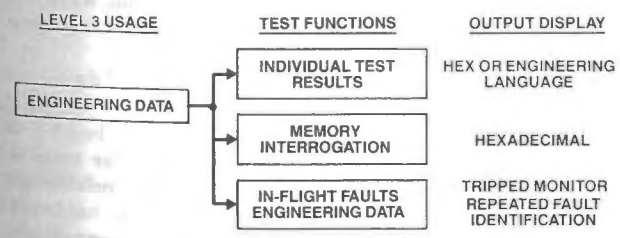


Figure 6. Engineering Data

Menu Selection

Once DFCS BITE is selected from the four FMS BITE subsystems, the operator is given the choice of selecting one of two menus. These are LINE MAINTENANCE and OVERNIGHT MAINTENANCE, which correspond to page 1 and 2 respectively. Line Maintenance consists of the fault isolation selections of IN-FLIGHT FAULTS and CURRENT STATUS and the system verification tests of LRU INTERFACE and LAND VERIFY. The OVERNIGHT MAINTENANCE page consists of selections for GROUND FUNCTIONAL TEST, RIGGING, and SENSOR VALUES, all of which are exercised when time is not as critical as when a 20-minute turnaround period should be achieved.

Ground Maintenance BITE Design

DFCS BITE is autopilot system-oriented covering sensors, interfaces, FCCs (Flight Control Computers) and surface servos, with fault isolation of prime importance. Because of the comprehensive A/P engage logic fault monitoring and multiple interfaces involved in autopilot operation, fault-isolation could be extremely tedious without BITE. BITE is designed to fault isolate and display on the FMC-CDU 93 different aircraft replaceable LRUs and components as quickly as possible. Automatic selection of the installed channel is done wherever possible to save decision time. When both FCCs are installed, BITE testing in both channels is accomplished simultaneously.

The BITE software is resident in each FCC which contains two processors. BITE software is isolated from flight critical software so that there is no interference with in-flight operation and recertification is not required for in-service BITE changes. CPU2 BITE software has been designed using general routines commanded by CPU1 making it practically insensitive to future changes.

BITE tests are divided into the following categories:

1. Quick tests - These require no maintenance personnel action other than an initial aircraft set-up.

2. Interactive tests - These require maintenance personnel action because sometimes this is the only way to check certain switches and displays.

3. Surface tests - These require surface motion to provide comprehensive aircraft testing.

Categories 2 and 3 may not be required if quick tests can uncover the fault.

Quick test identify failure within the following DFCS equipment or functional areas:

- Initial state of system logic inputs
- IRU 429 Bus labels and status
- Navigation Receiver self-test
- DADC 429 Bus labels and status
- A/T 429 Bus labels and status
- DME self-test
- Alpha vane interface
- LRRA
- Digital LNAV
- MCP 429 Bus labels and internal self-test
- N1 Pot
- FMC 429 Bus labels and status
- Flight Mode Annunciator interfaces
- Surface LVDT and Position sensor interfaces
- CWS transducer interfaces

- FCC power-up faults
- Autonomous and cross-channel busses
- Standard option and ADI differences
- FCC A/B index and synchronization
- Flap comparisons between channels

Interactive tests identify failures within the following DFCS equipment or functional areas:

- Mach Airspeed indications
- HSI indications
- ADI indications
- Flight Mode Annunciator displays
- MCP controls and displays
- Speed trim warning
- ILS Deviation warning
- Alt alert annunciator
- Barometer switching
- Disengage warnings
- Instrument switching

Surface tests identify failures within the following DFCS equipment or functional areas:

- Elevator Actuator/PCU
- Aileron Actuator/PCU
- Mach Trim Servo
- Stabilizer Trim System
- Dual Control

The LRU at fault is displayed immediately in English language after that test is completed rather than waiting for completion of the entire BITE testing. A complete diagnostic summary is then given after quick tests, interactive tests, and surface tests.

The following engineering modes have been extremely useful in BITE development to fully debug software, and test new aircraft for delivery. These modes are not normally utilized by the line maintenance personnel. However, the following are available to the specialist technicians:

- Test Results - A sample display is shown in Figure 7. Each display is capable of four sets of results on both channels. Digital readings which pass are not displayed. (Only selectable via engineering code.)

(200, 6R For Memory Interrogation, E, -6R To Exit)

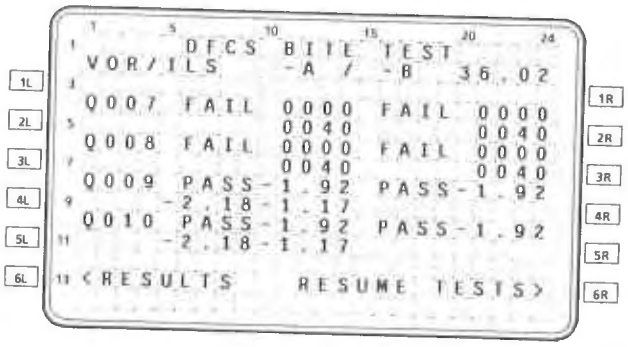


Figure 7. Test Results

- Memory Interrogation - A sample display is shown in Figure 8 showing a reserved display line for memory contents of the selected locations. (Only selectable via engineering code.)

(100, 6R For Test Results, E, -6R To Exit)

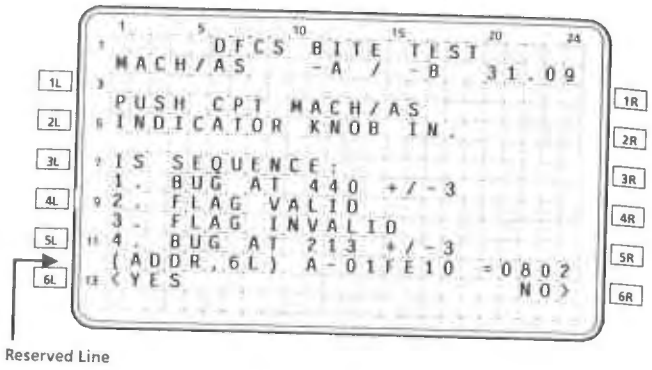


Figure 8. Memory Interrogation

- Ground Functional test select/delete alternate mode to run only the selected tests or to run all except the selected tests.

- Display of active options and program variations in English Language.

- Display of aircraft surface rigging procedures and readings. These consist of Elevator, Aileron, Flaps, Stabilizer, Neutral Shift, and Control Wheel Steering.

- Sensor Values - Displays the present values of sensor outputs in physical units.

In-Flight Monitoring Routine

In-flight monitoring is provided to check the basic systems health of the dual channel DFCS, dual MCP, and the supporting sensor signal validities. The internal monitoring is active full time whereas certain system monitors are only activated during particular flight phases or when a given autopilot mode is engaged.

Fault Reaction

Whenever a failure is detected by the internal or system monitoring, a fault reaction is processed within the computer units. Four major events take place: 1) system reconfiguration, 2) annunciation of the system failure, 3) activation of Maintenance Monitor Triggers, and 4) activation of additional Maintenance Monitor routines for certain fault types.

System Reconfiguration and Annunciation

The DFCS reconfigures by disconnecting the engaged channel or channels, preventing autopilot engagement, biasing out of view the Flight Director Bars on the Altitude Director Indicator (ADI), and/or switching over to the redundant system contained within the second autopilot channel. Annunciation occurs for loss of the following functions: Autopilot, Speedtrim, Mach trim, or Autotrim. Whenever the internal monitors have detected a failure, the ADI displays a computer flag.

Basic Health Monitoring

Detection of failures within the DFCS computer units is provided by means of extensive internal "continuous monitoring" and heartbeat monitors which are provided by hardware mechanization to detect loss of CPU operation. Power supply monitors are either dedicated hardware circuits or implemented via software routines.

Continuous Monitoring

Continuous Monitoring is provided within the DFCS for both internal operation and external system checks of actuation and supporting sensor/intercomputer signals.

Internal Monitoring

These tests are generally conducted by software (except as noted) for both the main and secondary processors.

- ROM
  - Background cyclic redundancy check.
  - A standard check-sum on the inner loop program and I/O processing storage.
- RAM
  - Background - read/write operation exercised. (All CPU writeable RAM areas tested over a given number of frames.)
- Processor Tests
  - All control and operational instructions, register operations, and various tests of routines such as: status logic, jump select, shift, overflow, multiplexers, instruction decode, and mask ROM test, etc.
- Ticket Check
  - Program flow is checked to ensure all modules are exercised.
- D/A - A/D Wraparound
  - Special variable test words are wrapped to check I/O processing and $\pm$ 15 Vdc power. All analog and digital outputs are also tested in this manner.
- DMA Tests
  - Parity, assess error, interrupts, input update rates, write protect, etc.
- Stimulation
  - All stim locations used at power-up are tested for zero.
- Heartbeat (hardware)
  - Monitor for a software generated pulse within a specified time window.
- Power Supply (hardware)
  - 28Vdc and 5Vdc monitors.

System Monitoring

The system monitors and monitors activated during dual channel operation are listed below:

1) Servo LVDT Common Mode
2) Sensor reasonableness, validities, label refresh rates, and parity
3) Servo Command Response Monitors
4) Mach Trim Command Response
5) Speed Trim/Autotrim Command Response Monitor
6) Surface Position Monitors (dual only)
7) Servo Command Monitor - active channel vs model (dual only)
8) Radio Altimeter - Fine/Coarse Comparator (dual only)

Power-Up Testing

In order to reduce exposure time, critical monitors are tested one time at power-up on the ground. Other testing is conducted to test the disengage functions, program option pins, and complete RAM and EPROM memory tests. These tests must be satisfactorily completed (14 to 15 seconds on ground and 2.5 seconds in flight) to allow DFCS functions to operate.

Maintenance Monitoring Routine

The Maintenance Monitor function is provided to store in-flight faults that have been detected which cause some processor reaction. The computer shall isolate the cause of a failure to an LRU. In order to reduce the number of faults that are stored, the scope of the monitoring is limited to loss of major functions, autopilot/flight director system disengagement, and/or inability to engage the autopilot.

Correlation with Cockpit Effects

Whenever a fault is detected within the system the Maintenance Monitor identifies the failed LRU to the line mechanic and correlates the identified unit with a particular cockpit effect. The cockpit effects that are utilized are as follows:

1) Unsuccessful autopilot engagement
2) Autopilot disconnect
3) ADI computer flag in view
4) Flight director bars "biased out of view"
5) Loss of Mach trim
6) Loss of speedtrim or autotrim

Fault Storage

Failures are stored in nonvolatile memory and separated by different flight segments. Ten flight segments are utilized in a push-down stack arrangement with the oldest data being lost as the latest data is required. A minimum of five faults per flight segment are stored. However, in practice a storage capability exists for up to 60 faults, dependent upon the type of faults that are being stored.

Flight Segments

In general a flight segment starts at airplane liftoff and ends on airplane landing. Exceptions to this occur for Flight Director takeoff and for internal "on ground" failures.

Maintenance Monitor Readout

Readout of the in-flight fault data is provided for the line mechanic through the Ground Maintenance BITE. Access is provided in the cockpit from the FMC-CDU with the failure data presented for the "last flight segment." At this level, the failed LRU and the cockpit effect is identified. Summary and more detailed data over the ten flight segments can be obtained by selecting the engineering level pages. At this level the following is displayed in English language: monitor that failed, nonvolatile memory location, and if the fault repeated during that segment. In this way the Maintenance Monitor can support rapid turnarounds or investigative troubleshooting by the engineering/specialist technician personnel.

Maintenance Monitor Design

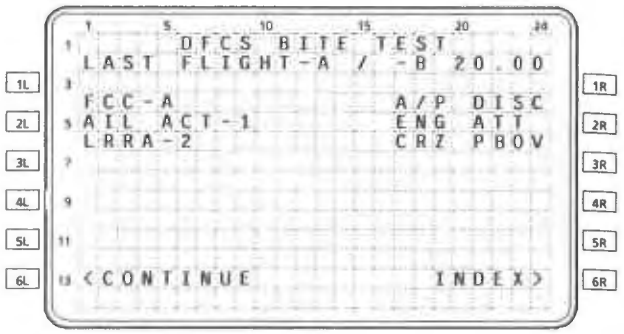
The applications (in-flight) program stores failure information into nonvolatile memory (NVM) for the following cockpit effects:

	No. of Tests
F/D FLAG - continuous monitor/power-up faults	122
A/P DISC - autopilot disconnects	64
ENG ATT - unsuccessful engage attempts	31
TGA PBOV - Pitch Flight director bias out of view	9
APP PBOV - Pitch Flight director bias out of view	9
CRZ BOV - Pitch Flight director bias out of view	23
TGA RBOV - Roll Flight director bias out of view	5
APP RBOV - Roll Flight director bias out of view	7
CRZ RBOV - Roll Flight director bias out of view	8
SPD TRM - Speed trim faults	5
MACH TRM - Mach trim faults	4
STAB TRM - Stab out of trim faults	3
MISC - Detection for loss of power	1

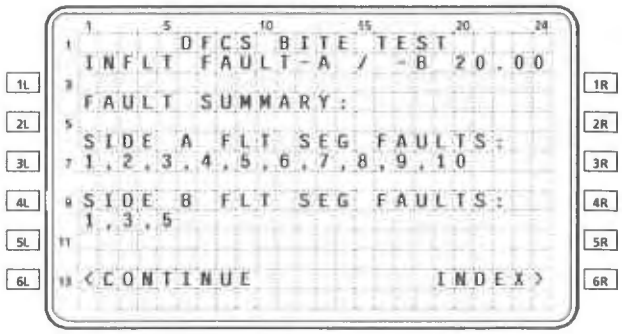
The above table also shows the number of different tests performed which can result in displays of different faults.

The coded information which is stored consists of LRU diagnostics, failure messages, cockpit effects, repeated fault, and analog data. On-ground BITE uses a system of table look-ups to convert the coded information to English language abbreviated messages. This information is displayed on the FMCS CDU. In the maintenance mode the displayed information consists of the probable LRUs at fault along with the associated cockpit effect for the last flight leg. Additional details are available to the maintenance engineer with the use of a coded entry. The additional details consist of a summary page of faults in the last 10 flight legs, the display of an extra line providing an English language abbreviation of the fault, a repeated fault indicator (R added to the end of the fault description line) and the display of the first fault word memory location. See Figure 9 for typical display formats. On-ground power-up faults are stored in the previous flight leg. If a dual disconnect occurs caused by tripping a servo performance monitor, suspected sensor valids are tested and the faulty LRU is stored. If there are no invalids, the LRU is "UNKNOWN" and the FCC stores analog signals for engineering use.

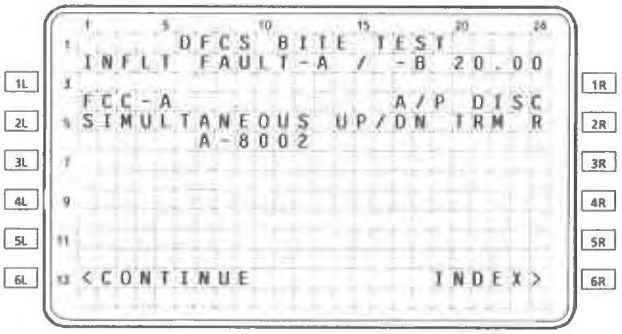
If analog signals are stored in the engineering mode, the address of the starting memory location (LOC Deviation No. 1) will be displayed under the failure message (see Figure 9). If memory interrogation is selected the LOC Deviation No. 1 data will be displayed in line 12 in hexadecimal format. Memory interrogation can then be used to retrieve the remaining nine signals for comparison of the LOC Deviation, G/S Deviation, radio altitude, vertical speed, and roll angle.



Last Flight Segment



In-Flight Fault Summary (300, 6R)



Engineering Data

Figure 9. Maintenance Monitor Display Formats

Experience With Maintenance Assist Functions

The BITE/Maintenance Monitor has been instrumental in detection of internal and external faults during the development phase of the DFCS. This ability has allowed for corrections to be made and subsequent flight test verification of the problem being resolved. With the 737-300 airplane, Ground Functional Testing is being conducted wholly by BITE and during the flight testing in-flight faults are obtained and utilized for debug. Specific problems identified: Internal, 1) DMA access errors, 2) pitch and roll surface synchronization difficulties; external, 1) IRU data bus short circuiting, 2) inability to null neutral shift sensor correctly and rigged 180 degrees out of phase, 3) limited Mach trim actuator travel.

On the 737-200 airplane a forerunner of the "federated" BITE system was developed utilizing a hybrid autopilot system. Similar system problems were resolved by BITE, and operator utilization has resulted in favorable comments. BITE has resulted in a reduction of the unconfirmed failure rates and has allowed the operators to quickly diagnose the cause of a system malfunction. This system is now in service both domestically and with foreign operators, and has proved to be most beneficial in removing "shotgun" maintenance techniques.

Introduction of new BITE systems requires technicians to be trained in their use, even though the design is made for ease of utilization. With the 737-300 BITE, training aids such as PLATO (Interactive Computer Aided Instruction) and simulations shall be employed to accomplish this task. The use of pocket insert cards showing the BITE menus have helped the line mechanic enter and operate the BITE routines.

It is therefore noted that BITE/Maintenance Assist Functions are being actively employed during the development and flight test phase of producing new airplanes. In-service utilization of BITE is reducing costly delays and unconfirmed removals, together with reducing the spare LRUs being carried in stores. The "federated" BITE system designed for the Boeing 737-300 allows for expeditious troubleshooting and for greater dispatch with a fully operating FMS. This will increase the maintainability of this new-version airplane over its predecessor the 737-200, which incorporated a less comprehensive BITE into the Performance Data Computer System (PDCS).

84-2709

INTEGRATED ON-BOARD MONITORING AND TEST CONCEPTS

Paul C. Jenkins, C.J. Ong

Collins Government Avionics Division
Rockwell International Corporation
Cedar Rapids, Iowa

Abstract

The application of multiplex buses for the interconnection and integration of avionic systems has positively influenced the implementation of test and monitoring.

Distribution of the on-board built-in-test and monitoring functions, with centralized control and display, is now readily possible. Additionally, the use of multifunction displays with software controlled display formats enhances reporting and fault diagnosis at several levels.

This paper takes a systems-level approach to the development of effective test and monitoring. A set of required support modules based on a layered functional architecture is discussed. A cost-effective, open-ended, design philosophy is proposed in terms of current and near-term capability.

Introduction

The title of this paper is very broad. Specifically, we discuss the integration of distributed monitoring and test within an arbitrary multiplex databus connected avionic system.

Systems Viewpoint. Historically, the specification and design of BITE and monitoring has been confined to individual LRUs or isolated subsystems. Where safety or operational necessity are major considerations, subsystem BITE/monitoring is given high priority.

The digital Autopilot and Flight Director System (AFDS) for the Boeing 767/757 is an excellent example (1). Within the AFDS flight control computers, 56% of memory contains maintenance and redundancy management software compared with the 27% devoted to airplane control. Such a high-end implementation must be contrasted with the inadequate BITE provisions in many existing LRUs. The advent of military standards and requirements for highly automated systems are leading to a minor revolution in system simulation, integration, verification, and testing.

However, no standards are generally applied for the application, initiation, and reporting of BITE/monitoring.

There is activity in progress to provide for some bottom level uniformity. The ARINC organization is working to establish military and commercial standards. The specification of standard military modules will necessarily involve standardized monitoring protocols (5).

The operation of BITE/monitoring is typically viewed from the top or the bottom. Attractive scenarios have been described involving graphic color displays, voice interaction, and other advanced techniques supporting crew or maintenance personnel in determining aircraft health.

The bottom level approach is based on LRU BITE fault isolation specifications. Sometimes these criteria are not clearly identified with system level performance. For that reason it has been notoriously difficult to verify actual performance against specification.

If splendid top level interactive techniques are employed based on false or misleading data then their value decreases. Crew and technician confidence in the BITE/monitoring system is crucial (6).

Realism dictates that when times get tough the BITE gets booted. Historically, it has occupied a low status in noncritical areas. At the systems level we must bridge the gap between customers' expectations and actuality.

It is necessary to develop a system-level, nonspecific, integrated, BITE/monitoring approach. Like all good system engineering it starts with an analysis of the requirements, an evaluation of current implementations, and the development of a descriptive model.

Military avionic systems using multiplex data buses have formed the major area of discussion. However, in developing the descriptive model, a large body of commercial aviation experience has been reviewed. There are many common areas.

Charles Kettering once said, "The only difference between theory and practice is that in practice you can't leave anything out." However, practical omissions have been noted in current BITE systems. Some of these would have been avoided if a sound theoretical process model had been used.

The remainder of this paper presents consensus observations on BITE implementation and then proceeds to present a strawman system BITE architectural model.

Information Sources

In developing the BITE/monitoring model we have drawn on the following resources:

Experience. Collins Avionics BITE design experience includes the KC-135 Fuel Savings Advisory System, A-10 Control and Display Unit, F-111 Avionics Management Program, Global Positioning System user equipment, Boeing 757 and 767 Digital Flight Control System and Engine Indicating and Crew Alerting System (EICAS), and the Lockheed L-1011 Digital Flight Control and Active Control Systems.

Study. Several recent study activities conducted by Rockwell-Collins Divisions have provided a body of BITE/monitoring concept analysis. The data drawn from large aircraft investigations such as the MD-100 and C-17 has shown significant requirement similarity with future systems projects such as ARTI and LHX. The presence of recurring features in terms of requirements, partioning, and operation reinforces the need for a systematic approach to BITE which is not vehicle specific.

Industry-Wide Survey. During the last eighteen months an industry/government survey was conducted in order to determine areas of consensus. The following factors were considered:

1. Usefulness and deficiencies in current BITE/monitoring implementations
2. Unfulfilled requirements and enhancements
3. Future developments

A BITE questionnaire was distributed to maintenance personnel at several major national and international airlines. Later some of the respondants agreed to participate in face to face sessions with our BITE survey team where they were able to elaborate on their comments. The survey team also visited Wright-Patterson AFB, Scott AFB, and McDonnel Douglas, St. Louis, for a military perspective.

Summary of BIT, BITE, and Monitoring Observations

1. The Need for Front End Emphasis. Historically, BITE has had less priority than operational systems. BITE circuitry was a large expense early in the program not only in terms of cost, but of LRU real estate. Consequently, it seldom performed according to specification. When 90 percent or greater fault detection was called for, often only 50 to 60 percent was realized. This leads to problems for the support organization.

Inadequate BITE results in false equipment pulls, cannot duplicates (CNDs), and retest OKs (RTOKS). Immediately, the number of required spares and maintenance times increase. Overall aircraft availability decreases. Resulting spare shortages mean grounded aircraft.

BITE must become a front end design characteristic and be developed along with the operational systems. Experience has shown that good front end progressive BITE can be valuable in testing and debugging during avionics installation (4).

2. The BITE System Must Be User Friendly. This was the most used expression in describing the problems with current BITE implementations. The system should be easy for the operator to use.

Multifunction displays are preferred as the interface for both maintenance technicians and crew members. Electromechanical display devices are confusing and result in poor system reliability. The BITE operator interface as well as the BITE system itself must be more reliable than the aircraft operational system. Otherwise, what can the maintenance technician do when BITE fails?

Test procedure manuals on the aircraft are considered unacceptable. A common recommendation was to display aircraft Technical Orders on the BITE information display system. Technical Orders also cause problems when fault isolation procedures do not match software diagnostic logic trees.

Having several different local methods of BITE display on the same aircraft is unacceptable, presentation must be uniform.

BITE systems should be highly automated, leading the operator through the decision making process.

Lastly, the "User Friendly" system must provide a mechanism for getting maintenance data off the aircraft "untouched by human hands."

3. The Scope and Operation of the BITE Must Be Comprehensive. A comprehensive maintenance plan must be developed. From this foundation integrated BITE specifications can be developed.

Preferred military integrated BITE system design would provide stand-alone capability. No flight line Ground Support Equipment (GSE) would then be needed at austere bases.

Integrated diagnostics policy must be considered when looking at military BITE requirements. This requires all weapon systems to incorporate capability to detect and unambiguously isolate 100 percent of the faults known or expected to occur. Three distinct disciplines are brought together: Automatic Test Equipment/Built-In-Test (ATE/BIT), Maintenance Aiding, and Maintenance Training.

4. Fault Detection and Isolation Must Be Accurate. Major problems with today's BITE methods include fault isolation to the wrong LRU and nuisance (clutter) faults. Many users suggested that fault balls not be used at all because of their erratic behavior. Once the technician no longer believes the BITE system and resorts to alternate methods of fault isolation, problems become more serious and fundamental design problems are ignored.

Because of higher avionics reliability, the technician is going to see fewer failures and will forget how to perform specific maintenance procedures. The BITE must be more accurate and comprehensive to compensate for this.

As avionic systems become digital and more reliable, bus and wiring faults become more prominent. This could result in more cannot duplicates (CND's) if not recognized and dealt with accordingly.

There is a need for front end realism when establishing fault detection/isolation requirements. Aircraft availability is severely reduced when support and maintenance tasks are specified for an aircraft with 95 percent fault isolation/detection requirement when in reality it is only achieving 50 to 60 percent.

5. BITE Should Be Embedded. The consensus was that the BITE design should be embedded in the avionics in lieu of a central system. This approach capitalizes on the inherent capability of the avionic system and designer expertise.

6. BITE Control/Display Should Be Centralized. In all cases, the users recommended that the line technician have one location for fault detection and diagnosis.

Alternate methods for fault data retrieval such as printer, magnetic tape, or telemetry were also recommended.

Development of a Generalized Architectural Model

Integrated BITE/monitoring design requires the development of a generalized model. Computer/computer dialog standardization has been facilitated by the development of the ISO-OSI with its seven-layer view of the communication process. Our strawman BITE/monitoring model takes a similar view of integrated BITE operation.

The objective is the identification of required functions both in current and future systems. BITE/monitoring interfaces are discussed in terms of system resources rather than actual hardware characteristics. The model serves:

- To identify minimum system implementation
- To provide a basis for preplanned enhancement
- To facilitate system BITE design
- To identify common operations.

A Layered Approach to BIT, BITE, and Monitoring

Overview. The model illustrated in figure 1 applies to two aircraft status levels.

1. Operational. Applies to the preflight, inflight, and postflight operational phases. Here, the BITE/monitoring system principally functions to support reconfiguration and alert the crew to failures detrimental to avionics system performance. A secondary function logs failure event data for later analysis.

2. Diagnostic. Military services are increasingly interested in two-level maintenance. Correctly designed diagnostic BIT/monitoring can employ the extensive computational capability which exists in modern military aircraft together with a minimum of special test equipment to provide unambiguous fault isolation. This presupposes that an adequate BITE infrastructure exists. The aircraft multifunction controls and displays can be employed in a diagnostic mode to provide a high level of interactive technician support.

The model is not directly related to physical hardware except at the lowest layer. It identifies the functional elements required without specifying their location. This is becoming more important as the mapping of functions to LRUs is altered by VHSIC applications. Indeed, future architectures will be somewhat dynamic in their functional allocation. Functional failure which is most important in the

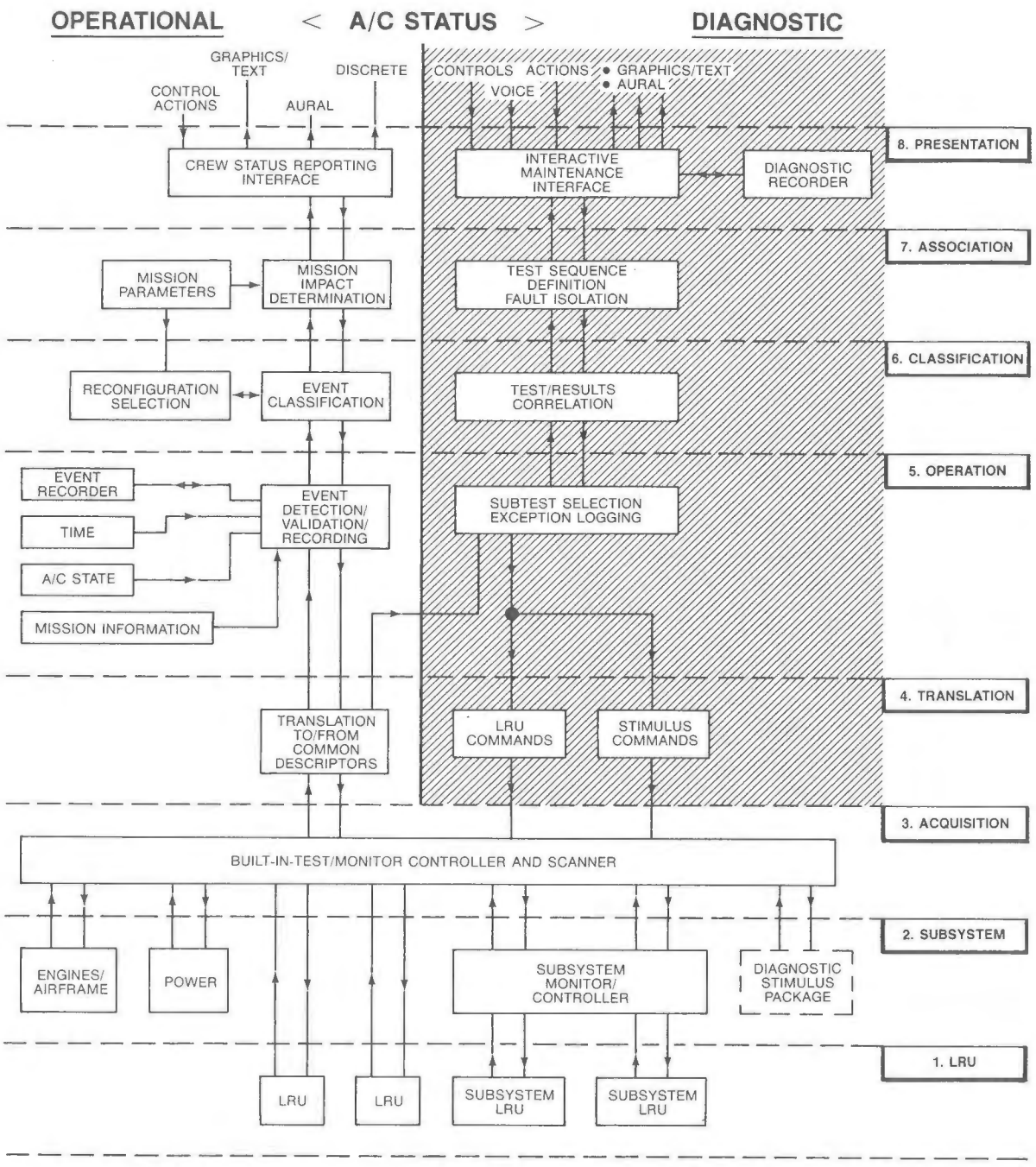


Figure 1.

operational context is becoming less tied to physical unit failure which is the major diagnostic interest.

Layer by Layer Description

The architectural model shown in figure 1 consists of eight layers. Up to layer 3 the functional elements are similar for both OPERATIONAL and DIAGNOSTIC operation. It is, therefore, appropriate to start the description at the bottom and proceed upward.

Foundation Layer Functions. The foundation of the BITE architecture is provided by the LRU, SUBSYSTEM, and ACQUISITION layers.

1. LRU. The lowest level contains the Line Replaceable Units (LRUs). Currently these are ATR boxes; in the future they will become modules as in the Pave Pillar and LHX concepts. Each complex LRU should contain sufficient BITE to allow unambiguous fault detection and isolation to the SRU level. Currently some simple LRUs exist where extensive BITE is not cost effective. However, as microprocessor based systems become ubiquitous, BITE overhead will become a minor recurring cost factor. For new equipment it is important that the LRU designer include BITE requirements at the outset. It is particularly important in interface design to provide for integrity checking of input and output circuits as far as possible. Many false box removals are caused by faulty wiring. LRU level BITE is continuous (C-BIT) or initiated (I-BIT). Since I-BIT interferes with LRU function it should only be used diagnostically except when automatically run at power up.

2. SUBSYSTEM. Some subsystems are sufficiently self-contained or have special characteristics which merit some level of autonomous

BITE. Currently Flight Control Systems use subsystem BITE/monitoring for obvious reasons. In the future as functional density increases advanced subsystem BITE will become more common. It will also become increasingly mission-critical in new highly automated systems.

An approach to Integrated Communication Navigation Identification Avionics (ICNIA) has been described (2) which features mission phase dependent reconfiguration. This level of subsystem capability requires extensive system level support, particularly with respect to functional priority.

Both C-BIT and I-BIT can be employed at the subsystem level. This is illustrated by the diagnostic stimulus package which includes any continuous test messages to establish the integrity of the inter-LRU communications media. It also covers special purpose test equipment. In both military and commercial avionics there is increasing pressure to build this equipment into the airplane rather than rely on suitcase testers.

3. ACQUISITION. This layer manages the acquisition of BIT/monitoring data from subsystems and LRUs. It also controls the initiation of I-BIT and the operation of the diagnostic stimulus package. In the airline world using ARINC broadcast buses, the acquisition layer is a physical entity such as the Aerospatiale Maintenance Interface Unit (MIU) (3). In the military domain, extensive use of MIL-STD-1553B multiplex data buses both removes the need for acquisition hardware and facilitates system BITE implementation. In such systems LRU intercommunication is managed by a single bus controller which usually operates according to a deterministic transaction schedule. This schedule includes diagnostic messages which are employed by higher layers to continuously assess the performance of the system.

Status Dependent Layer Functions

4. TRANSLATION. This layer contains and manipulates LRU and subsystem data which is implementation specific. It contains a database of BITE/monitoring reporting and initiation conventions.

Operational Role. In general, no standards are currently in effect controlling the reporting protocol for BITE/monitoring. When a standard communication protocol is used, some boxes report BITE using binary codes while others use strings of ASCII characters. Even when descriptive strings are used, there is little uniformity. Operationally, the translation layer converts all LRU and subsystem reports into a standard system-reporting format. Ideally, this contains not only failure information but also notes the consequences of the logged failure at the LRU level.

Diagnostic Role. Diagnostically, the translation layer serves to expand a high level test definition into a set of specific LRU and diagnostic stimulus commands. In multiplex systems, the central management function of the bus controller can be employed to mimic sensor inputs or crew actions and to collect the results.

5. OPERATION. The operation layer is the highest to be concerned with the intimate manipulation of BITE/monitoring processes.

Operational Role. Operationally, this layer determines the validity of malfunction reports. Ideally, time, mission-phase information, and associated subsystem monitoring (eg power) are used as validation criteria. Significant events — those qualifying as real — are referred to the next highest level together with any supporting information from the translation layer. The operation layer also provides the BITE/monitoring recording function. Significant events are recorded with time tag, mission tag, and other context information. Future recording media will provide enormous amounts of storage. This can be used to store information snapshots for all events including power, engine/airframe, crew input data and crew display formats. However, this data must be structured to allow for access by different categories of users.

Diagnostic Role. The operation layer executes a test defined by the upper layers. This may consist of a set of related actions or subtests. Information from the translation layer is used to identify exceptions associated with subtests. Action/reaction data are referred to the upper layers for analysis.

6. CLASSIFICATION. The classification layer operates on events in an abstract manner. It is not concerned with the mechanization of BITE/monitoring, only with the significance of information derived from it.

Operational Role. Validated events which could directly impact the avionics configuration, the margin of mission success, or which need to be forwarded to the crew are input from the operation layer. Event data together with supporting information from the translation layer are used to classify the severity of the failure. Loss of system resources actuates a reconfiguration process which uses mission phase and parameter data to retain vital capability. In military aircraft, workload reduction dictates that most of this process should be crew transparent.

Diagnostic Role. High level definitions are passed to the lower layers for execution. Results are used to create a resume of events and consequences for higher level processing.

7. ASSOCIATION. This layer provides significant workload reduction by incorporating "intelligent" decision making processes previously delegated to the crew or maintenance personnel. Operationally, crew consultation would be minimized. Diagnostically, extensive technician support and engineer consultation services would be provided.

Operational Role. Information on significant events is used together with the mission parameters database to determine mission impact. This relates not only to the current mission phase but also to future phases. In military aircraft the crew would be advised on potential functional degradation and given an estimate of mission success.

Diagnostic Role. As far as possible, the association layer provides the automatic sequencing of tests necessary to provide unambiguous fault isolation. Avionic maintenance should involve a single technician with minimum manual intervention. Specialist engineers would be provided with detailed background information and the ability to define and execute new test sequences.

8. PRESENTATION. The facilities of the glass cockpit are employed to provide the appropriate level of support and information to the crew and maintenance personnel.

Operational Role. Alert conditions would employ discrete and aural indicators with crew selection of status displays. Preemptive displays would not be used, though caution/warning fields could be reserved in display formats. Top level information is provided concerning loss of mission-related aircraft functional capability. Data relating to LRU/SRU failure is not normally presented to the crew. Displays are graphic with minimum top level text. Crew decisions regarding reconfiguration should be presented as multiple choice summaries if absolutely necessary.

Diagnostic Role. Two elements are present in the diagnostic presentation layer:

a. Maintenance alert data transfer
b. Interactive diagnostic support

The first element is provided by the event recorder maintained by the operation layer. The recorder may be a physically removable entity or a distributed memory driving some kind of data link. It is important for this data to be obtained in a manner which does not interfere with mission preparation or require that the aircraft be fully powered. Ideally, a second level of event recording would allow full simulation of all aircraft parameters including displays and crew actions associated with a logged event. This would facilitate the resolution of fundamental problems due to residual system software bugs.

The second element of diagnostic support should be provided by the existing cockpit displays and multifunction controls. This type of interface system is needed to "bridge the gap" between the inexpert user (line technician) and the very sophisticated systems to be maintained. A diagnostic recorder function is provided by the PRESENTATION layer. This could be the same physical unit as the OPERATION layer event recorder. However, it is used here to provide a removable maintenance session log including system prompts, operator actions, test results, etc.

Background material relating to the operation of the ASSOCIATION and CLASSIFICATION layers could be included for later review by specialist personnel where necessary. The diagnostic recorder also serves as a source medium for test sequence definition updates prepared off aircraft. It can also be used to update a variety of other BITE system data bases.

The Future

Two-Level Maintenance Concept. Two-level maintenance is a prime item within all of the Air Force system commands. Cost is the driver away from the present three-level concept. The intermediate level shop has become cost prohibitive due to the quantity of equipment within each shop and number of locations required.

Pave Pillar. The Pave Pillar objectives are to:

1. Develop and demonstrate the next generation integrated avionics system architecture which will enable significant improvements in Availability, Cost of Ownership, and Mission Effectiveness.
2. Eliminate Avionics Intermediate Shop (AIS) — The approach is to improve reliability and maintainability by reducing the number of cables and connectors, design fault tolerant systems, modularize, and implement integrated test and maintenance concepts.

Integrated Diagnostics. Integrated Diagnostics is receiving increased attention throughout the Office of Secretary of Defense (OSD) and the Military Services. It is defined as a structured process which maximizes the effectiveness of diagnostics by integrating pertinent elements. These include testability, automatic/manual testing, training, maintenance aiding, and computer-aided engineering.

Future Avionics Design Needs. The Military Services are very serious about improved aircraft availability. Key future avionics needs include:

1. Minimum maintenance, often by lower skill levels
2. Design for "soft" failures
   a. Automatic software reconfiguration to bring up hot spares
   b. 1000 hours (minimum) before any maintenance required
   c. 5000 hours (objective) before any maintenance required
3. Partition for ease of maintenance and logistics
   a. Minimize number of common module types
   b. Use common modules across all subsystems
4. Improved diagnostics and self-test/built-in test
   a. Self-Test (ST) end-to-end on functions across LRUs.
   b. Built-In-Test (BIT) internal to each module and LRU
   c. 98 percent failure detection to Shop Replaceable Unit (SRU) or Module
   d. Less than 1 percent Cannot Duplicate (CND) and Retest OK (RTOK)
   e. Vital maintenance data must be recorded.

5. Exploit technology advancements (eg, VHSIC).

Objective: More reliable, available, and supportable weapon systems.

6. Design and partition for minimizing maintenance and logistics.

Objectives: Soft failures, nonconfigurable subsystems, self-test, Built-In-Test to permit two levels of maintenance.

7. Promote rapid deployment and mobility (Bare Base)

Objectives: No avionics maintenance required for periods of 90-120 days.

8. Automate maintenance data recording and procedures

Objectives: Automate all data inputs, extractions, and tech orders; common data base for vertical testability.

Conclusions

Avionics BITE/monitoring systems have proved to be very variable in their usefulness. At their best they can facilitate all aspects of system development and operation. At their worst they can adversely impact aircraft availability by providing misleading information.

It is essential to take a systems level approach to BITE with front end emphasis on LRU embedded design. The use of microprocessor based hardware has simplified some aspects of BITE implementation. Future soft technologies will serve an analogous function at higher levels. Currently a good BITE system would contain well-designed LRU, SUBSYSTEM, ACQUISITION, and TRANSLATION layers. These present no major implementation problems.

The data logging function of the OPERATION layer would be limited by the capacity of current storage media and the extent of the validation processing. The functions of the CLASSIFICATION and ASSOCIATION layers assume a level of intelligence difficult to achieve without excessive cost and hardware overheads. Nevertheless, this degree of capability will be essential, in fact, mission critical in advanced new programs. Modern aircraft contain excellent on-board PRESENTATION layer services in terms of display systems and multifunction controls. Current data storage technologies, particularly optical disks, provide excellent media for the interactive display of maintenance information and schematics.

It is apparent that the intelligent CLASSIFICATION and ASSOCIATION layers are the weakest links in a truly comprehensive BITE system. Fortuitously, they are also the most abstract layers, manipulating events according to an externally defined context.

The nonspecific nature of these intelligent layers and advances in AI technology hold the promise of standard implementation with consequential cost reduction. Pragmatism therefore dictates that when funding and resources are scarce, emphasis should be placed on a solid system BITE foundation. Upper layer functions will be facilitated by technological advances but cannot compensate for inadequate LRU and SUBSYSTEM layer implementation.

If the retrofit of smart system BITE is contemplated for existing multiplex bus based avionics systems, these factors must be addressed.

1. What consideration was given to BITE during initial design, ie, will the foundation layers support it?
2. What deficiencies exist in the design and can external compensation be used to correct them?
3. What are the cost trade-offs, ie, will the retrofit be cost effective?

Unless P^3I has been a design criterion, it is doubtful that retrofitting a system level integrated BITE/MONITOR can be accomplished in a cost-effective manner. The redesign of hardware and software would be prohibitive. However, if a comprehensive avionics upgrade is being considered such as the B-52 OAS, then careful planning and design can result in a capable BITE system implementation.

Acknowledgement

The authors wish to thank John L. Wiley and Charles H. Scott for their invaluable assistance in the preparation of this paper.

References

(1) Peter Bradbury, "Digital Autopilot Monitoring and its Applications to the Boeing 767/757 Aircraft," Fifth Digital Avionics Systems Conference, 1983.

(2) Frank W. Smead, "Utilizing Basic ICNIA Capabilities to Improve Mission Availability and Reduce Pilot Workload," Fifth Digital Avionics Systems Conference, 1983.

(3) F. Coustalat, "Maintenance Centralized Data System (MCDS)," Unpublished Draft Paper prepared for AMC/AEEC TG 110 Bite Subcommittee Meeting, 1984.

(4) A. J. Dandekar, "CRT Displays in Air Transport Cockpits....Experience to Date," Fifth Digital Avionics Systems Conference, 1983.

(5) R. Gilbertson, "Development of Consensus Modular Avionics Standards," NAECON, 1984.

(6) Richard A. DeSipio, "Avionics Maintenance — A Perspective," Fifth Digital Avionics Systems Conference, 1983.

AVIONICS HARDWARE DESIGN FOR TESTABILITY

Barry L. Ferrell* and Steven L. Over*

General Dynamics/Fort Worth
Fort Worth, Texas

Abstract

The need for improved avionics design practices and on-board test techniques and strategies is discussed. System testability design problems are reviewed, along with guidelines for eliminating or reducing the causes of false failure indications and incorrect equipment removals. The effect of technology growth on on-board testing is discussed, and potential on-board test methods are described which could be employed to overcome the gap between technology and testability. An on-board test strategy is described which concentrates test resources at the removable assembly level or below for obtaining high levels of fault detection and fault isolation.

Introduction

Two supportability goals are reduction in support costs and higher equipment availability. One way of decreasing support costs is to reduce the amount of support equipment necessary for base-level maintenance. The economic necessity to repair equipment at the base shops is driven by the costs of spares for systems implemented with large, expensive, high failure rate black boxes (1). The design solution for attaining this goal has been identified as a design having smaller, less expensive, highly reliable modules - the equivalent in size to today's circuit cards. The second goal of higher equipment availability may be aided by fault tolerance, accomplished by graceful degradation, reconfiguration, or substitution of a "hot spare" into the system functional group. Both goals pose a challenge in the area of testability design because, for a modular architecture to be supportable and fault tolerant, the system must have the capability for real-time fault detection and fault isolation to the reconfiguration level. This must be achieved with greater accuracy than current systems are capable of isolating faults to the black box level.

A corollary to these requirements is the necessity to reduce or eliminate the incidence of false failure indications and incorrect equipment removals in avionic systems. A system in a continual state of reconfiguration, or one that requires multiple module removals per failure caused by incomplete fault isolation does not meet the stated goals.

These capabilities can be achieved by a system-wide approach to design of on-board testing. This includes improving system inherent testability design, implementing new testing techniques, and employing a comprehensive test strategy.

System Design for Test

Dealing With the Known Problems

Numerous programs have been initiated by the various branches of the armed forces to discover and correct the causes of false failure indications and incorrect equipment removals. To improve system testability performance, the system design must include corrections for known problem areas. Described below are some general problems of today's avionics systems, a discussion of potential solutions to each, and suggested implementations of the solutions.

Intermittent failures. Normal mate and demate actions on electrical connectors can cause corrosion and/or loss of retention capability, which results in system failure. The occurence of these faults is often intermittent, thus usually not duplicatable, resulting in frustrating, time-consuming maintenance. Minimizing the number of electrical interconnectons between modules will result in fewer cases of intermittent connector failure. This may be accomplished by functionally partitioning the design such that minimum data transfer is necessary between modules and subsystems.

Another potential source of intermittent failure is the sensitivity of electronic components to variations in power, temperature, humidity, vibration, and noise. The majority of errors in modern digital systems arise in memory components and communication paths. This type of fault often clears after system reset, power cycling, or during retest at a later time.

The effect of these intermittent failures

* Member IEEE

can be reduced by providing error detection and correction capability for memory and digital data transfer. Error detection and correction may be performed by employing Hamming or cyclic redundancy checks (CRC). These may be used to detect, or detect and correct single or multiple bit errors. Another method of recovering from intermittent failures is to employ automatic system reinitialization and fault filtering (i.e. built-in-test retry), where practical, to ensure a hard fault exists prior to system fault annunciation.

Discrete and Analog Data Transfer. Most current aircraft avionics designs use dedicated discrete and analog data transfer lines. The responsible test controller usually does not know the correct state or level of its dedicated discrete and analog data. Additionally, discrete and analog data lines from subsystem to subsystem are generally untested at either end. This uncertainty of correct state, in the event of a failure, leads to delayed fault detection, which results in fault propagation and ambiguity in fault isolation. An ideal design solution tests the transmitter, interconnect and receiver of discrete and analog signals. This is accomplished by performing signal injection and wrap-around testing techniques. Load sensing on discrete or analog outputs may be used to detect opens and shorts in wiring harnesses. Where analog output from sensors is required, the sensors should be designed to respond with a known output upon stimulation by the test controller.

Initiation and Operation. Multiple switch actions are generally required of the operator for system initiation and operation. An error in sequence or input data values may cause a system fault indication at a later time and is often attributed to hardware. Automatic system initiation by insertion of a portable data transfer medium containing all system initiation data will help reduce system errors caused by the operator.

Functional Partitioning. Poor distribution of functions between components often causes ambiguous fault isolation. An avionics system should be composed of modules and sensors whose functional identification is clear cut and whose interface is easily monitored. A method of obtaining this functional identification is to break the subsystem into circuit modules so that one or more whole, testable functions appears on each module. The result of this process is loosely coupled modules which perform semi-autonomous functions and exchange minimal control information.

Fault Annunciation. Current system and subsystem testing is generally controlled and reported through a single communication terminal with no alternate fault reporting means. Thus, no fault reporting capability exists in the event of controller or terminal failure which disables communication. A solution to this problem provides a dedicated fault reporting channel supplemented by fault reporting on the primary communications channel. These two channels should have different controllers so that failure of one would not inhibit a failure report. Another problem with current system designs is the case of the system or operator incorrectly identifying the source or location of a fault. In addition, "errors of omission" are a frequent occurence where the fault record is incomplete. One solution to the source identification problem is to include a fault management function for resolution of ambiguities. A solution to the omission problem is to provide a data link to communicate system fault information to the ground. Another solution is an on-board non-volatile fault log.

Technology Growth

Self-testing techniques in today's avionics vary from function to function. While many of these techniques are familiar and in wide use, the on-chip testing techniques are relatively new and rare. Integrated circuit technology growth has outstripped self-testing techniques and outdated testability verification methods. An overview of this effect is illustrated in Table 1.

Until the early 1970's, most fault detection/fault isolation (FD/FI) techniques dealt with the detection of "hard" or "stuck-at" faults (2, 3). These techniques primarily relied on periodic software testing to detect faults. In a typical avionic system, there is a high percentage of intermittent failures. The problem with relying on testing for intermittent faults is that the percentage of operating time spent executing test procedures is quite high if a high percentage of intermittent faults must be detected.

Another significant problem with trying to test some of the very high density integrated circuit chips is obtaining adequate fault coverage with a reasonable number of test inputs. An analogy to this problem is the off-equipment testing of today's circuit cards which are tested with hundreds or thousands of input test patterns. More often than not, many clock cycles are required to propagate a fault to the output connector. If large integrated circuits were to require this type of testing on-aircraft, the operating system could not tolerate the test delay in pattern processing.

Time Frame	60's	70's	80's	90's and on
Level of Technology	Analog, Discrete Digital	SSI, MSI, LSI, Some analog	Mature LSI Early VLSI/VHSIC	Mature VLSI/VHSIC
On-board test capability	Monitoring of critical functions	Self-test resident in software, hardware	Self-test resident in software, hardware	System-wide test strategy
Implementation Technique	Anomaly sensors, Voting schemes, Pilot observation	Mostly software controlled testing at subsystem level	Early implementations of on-chip testing. Majority of testing still accomplished by software at subsystem level	Integrated circuit level self-test imperative. Complete self-test must be accomplished at the replaceable assembly level
Specification Verification Method	N/A	Desk top analysis, Fault insertion testing	Desk top analysis, Fault insertion testing becoming extremely difficult due to chip complexity	Verification must be accomplished as part of circuit design
Level of FD/FI	Approx 50%	Approx 90%	Approx. 90%	98% Desired

Table 1 Technology Growth vs Build-in-Test Capability

Fault Detection Techniques

A very high degree of fault coverage may be obtained by monitoring the system with hardware error checkers which remain active during normal system operation. Use of error checkers enhances the system's ability not only to detect stuck-at and intermittent faults, but also allows the system to detect single or multiple faults. This technique, however, lends itself well to testing only portions of the logic circuit and may leave other portions insufficiently monitored (this is especially true of control logic).

There are several good test techniques that have been developed during the past few years to combat the IC testability problem. To name some specifically: Level Sensitive Scan Design (LSSD) is a patented approach used extensively by IBM (4); variations of Signature Analysis (i.e. BILBO's, Microbit, etc.) are used by some integrated circuit designers (5, 6); and combinations of both techniques are employed by still other IC designers (7).

These techniques, however, tend to make the integrated circuit more observeable only when executing function-interruptive testing routines, and are not generally useful for FD/FI concurrent with normal operation. In general, these approaches either leave areas of the circuit insufficiently monitored or they interrupt normal operation. The system designer is either forced to include enough redundant hardware so that the output of the hardware in question may be "voted" on, or a combination of less desirable techniques must be employed to provide acceptable levels of FD/FI for circuit types which are not easily tested by other means.

The following three techniques will provide additional capability for concurrent FD/FI. The first technique employed is the use of error detection and correction (EDC) at the module level. This technique would detect and correct most faults (including intermittents) on the main data path and in memory on each module. The second technique would require identification of typical faults that could occur which would not be detected by the EDC mechanism. The test designer would then develop test routines for these faults which could be executed by a dedicated Module Test Controller (MTC). All test routines necessary to test the module would be stored on the module. Prompts for these test routines could then be embedded in the host CPU's microcode or operating system, which would allow the MTC to test inactive hardware or to time-interleave test routines with the normal operation of the module. This would allow the testing function to be performed while remaining transparent to the applications programmer.

A third technique which may be employed is the selective duplication or use of already duplicated circuitry to perform periodic or continuous comparisons of outputs.

To achieve very high levels of fault detection, each system function, beginning with the integrated circuit level of assembly, should be analyzed to determine the combination of test techniques which has acceptably small

impact on hardware complexity while providing the desired test coverage. This is no simple task, but there is a high probability that this process will be integrated into the computer-aided engineering systems generally in use. The integration of design rules and techniques for on-board test provisions into computer aided design systems holds the promise of actually producing systems "designed for testability."

Fault Isolation Techniques

If proper functional partitioning is not employed during system design, problems which inhibit good fault isolation arise. In architectures which are not properly functionally partitioned it is difficult to unambiguously identify the source module responsible for the release of faulty information due to the unstructured exchange of control and data information. Designing the system with proper functional partitioning enhances good fault isolation because it specifies a structured interface with a well defined protocol which can easily be monitored for irregularities. Regularity of interfaces reduces the amount and type of fault isolation to be done. Proper functional partitioning, however, will not entirely solve the ambigious fault isolation problem.

One possible approach to enhancing the fault isolation process would be to require the system designer to identify error syndromes (signs or symptoms that occur together which characterize a particular hardware failure) where ambiguous isolation can occur, and then place error checking hardware in locations to monitor these syndromes. Once a given syndrome is detected, the FD/FI process is performed by a straightforward interpretation of the syndrome. A shortcoming of this approach is that it assumes each ambiguously isolatable failure has at least one unique error syndrome.

Testability Verification

At present, testability aspects of avionic equipment are measured 1) by desk top analysis, 2) by fault insertion testing, and 3) by computer fault modeling. Desk top analysis continues to be a good "first look" at equipment testability but too often is viewed as definitive when it is not. Fault insertion testing, while providing a "warm feeling" for equipment capability, is limited to a small class of faults if the test is to be non-destructive. Computer fault modeling is usually reserved for the generation of test patterns for off-equipment circuit card testing.

The first two methods are becoming less accurate due to increases in integrated circuit complexity. For systems employing complex IC's, the test engineer can no longer provide a timely determination of test coverage for a test he has generated. Fault insertion testing, where faults are limited to those inserted at the device interface, becomes increasingly unrealistic as integrated circuit complexity increases.

The introduction of new software for test pattern generation and testability scoring has increased in recent years and use of these programs is growing. Efforts in the industry are currently underway to integrate testability design and verification techniques into computer-aided design systems. Additional efforts should be made by integrated circuit manufacturers to identify the actual test coverage provided by on-chip test features. This information is necessary for generating supplemental tests to exercise the portions of the IC left untested by any on-chip techniques. For confidence in the on-board test capability of a system, on-board test must be elevated to the status of any other primary system function and its effectiveness must be measured as an integral part of the design effort.

Recommended Diagnostic System Configuration and Test Strategy

The recommended configuration of the diagnostic system is a testing hierarchy, as depicted in Figure 1. Fault detection should be accomplished at the lowest levels of the hierarchy, thus allowing the upper levels to control system resources and monitor system capabilities.

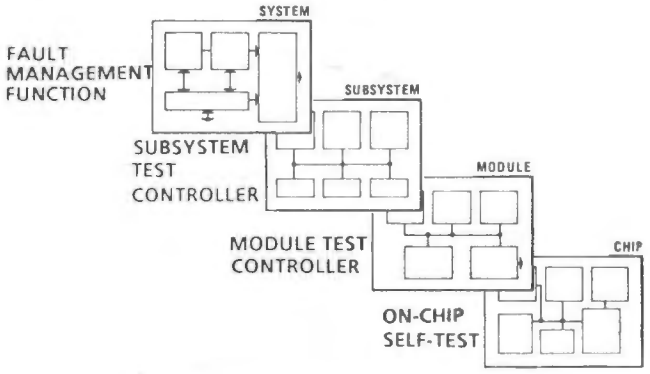


Figure 1 Diagnostic System Configuration

The test strategy recommended for avionic systems is a comprehensive system-wide approach. Testing should begin with the smallest component capable of self-testing and extend outward. Integrated circuits capable of self-test should assess their status and report to the module test controller (MTC). After the IC and functional groups are tested, the MTC should complete module level tests by performing module interconnect integrity tests and by testing components with no inherent self-test. At the subsystem level, the subsystem test

controller should perform subsystem interconnect integrity testing and it should poll module status provided by each MTC. At the system level, the fault management function should poll subsystem status and perform system interconnect integrity testing. This test approach will provide for fault isolation concurrent with fault detection for a majority of module faults.

A test strategy of this nature can provide high levels of fault detection and fault isolation with acceptable impact on system hardware. This concept, in concert with the system partitioning strategy improvements previously discussed, will help to reduce the causes of false failure indications and incorrect equipment removals. This concept also makes the most effective use of the on-chip test capabilities of the next generation of integrated circuits. This test strategy can help provide the means to achieve the desired goals of modular avionics and fault tolerance.

Conclusion

The need for improvement in avionic system testability design and effective on-board test capability for present and future avionics systems is clear. The provision of this improved test capability is the product of a design effort spanning each portion of the system. The design effort must include design for testability in every facet of the design and cannot treat on-board test as an afterthought. The gap between technology and testability can and must be overcome by careful selection of on-board test techniques. Function interconnection, component selection, operation, and test strategy must be considered on a system-wide level to provide effective fault detection and fault isolation. At the same time, false failure indications and incorrect equipment removals must be minimized.

REFERENCES

(1) S. J. Abraham, "The Relationship Between an Advanced Avionic System Architecture and the Elimination of the Need for an Avionics Intermediate Shop (AIS)," AUTOTESTCON '83

(2) M. A. Breuer, "Testing for Intermittent Faults in Digital Circuits," IEEE Trans. Computers C-22, P. 241-246: (1973)

(3) S. Kamal, "An Approach to the Diagnosis of Intermittent Faults," IEEE Trans. Computers C-24, pp. 461-467: (1975)

(4) N. C. Berglund, "Level-Sensitive Scan Design Tests Chips, Boards, Systems," Electronics, p. 108-110: March 15, 1979

(5) P. P. Fasang, "Circuit Module Implements Practical Self-Testing," Electronics, p. 164-167: May 19, 1982

(6) P. P. Fasnag, "Microbit brings self-testing on board complex microcomputers," Electronics, p. 116-119: March 10, 1983

(7) D. Komonystsky, "Synthesis of Techniques Creates Complete System Self-Test," Electronics, p. 110-115: March 10, 1983

REMOTE MAINTENANCE MONITORING USING A DIGITAL DATA LINK

Drew Dowling*
Richard A. Lancaster

ARINC Research Corporation
Annapolis, Maryland

Abstract

The purpose of this paper is to present a concept for upgrading the military aircraft maintenance approach in the future. The evolution of digital avionics in both military and commercial aircraft is creating changes that affect today's approach to maintenance. Commercial aviation has made significant progress in the direction of maintenance monitoring using a digital data link. This paper presents the status of maintenance-monitoring efforts within the commercial airlines and, recognizing the differences that exist between the military and commercial application, proposes an aircraft maintenance concept for the military in the 1990s.

Background

The sophistication of new aircraft systems currently being developed suggests that a new maintenance approach may be needed. The employment of Built-In Test (BIT) in avionics Line Replaceable Units (LRUs) to isolate faults can reduce the organizational maintenance activity to replacing designated components at a low-skill level. The isolation of faults not identified with BIT can typically involve sophisticated and expensive Automatic Test Equipment (ATE) and highly skilled technicians or engineers. With the trend toward increased packing density on printed circuit boards, and in turn, on Shop Replaceable Units (SRUs), the cost of these units is escalating. All of the above factors contribute toward potentially high maintenance costs in the Avionics Intermediate Shop (AIS). In an austere budget environment the maintenance activity is subject to budget cuts, resulting in reduced maintenance capability. This situation presents a critical problem in an environment where command emphasis is placed on operational readiness. In response to this problem, operational availability projections have become increasingly important to systems under development; however, this is often translated into achieving improved reliability of the system. The maintenance approach has a significant influence on the downtime, and extended periods of downtime will affect operational availability more than poor reliability.

Before proceeding with the discussion of a maintenance concept for the 1990s, it is appropriate to present some of the concepts that characterize current military aircraft maintenance. Flight operations, maintenance, and logistics are separate entities within the military. The AIS at wing level supports the operations of the wing, and the logistics or supply activity supports the AIS. Each of these activities (operations, maintenance, and supply) operates independently through the use of separate communications facilities.

Each operational aircraft wing is self-sufficient with intermediate-level maintenance capabilities. Avionics maintenance capabilities include extensive automatic test equipment to fault-isolate both LRUs and SRUs. The Air Force intermediate-level maintenance facility currently requires 4500 square feet of air-conditioned space; therefore, dispersal of the maintenance facility to forward operating bases presents major problems. With the increased sophistication of aircraft in the 1990s, the Air Force intermediate-level maintenance shop may become even larger, more costly, and less mobile. As a result of these factors, it is the objectives of this paper to suggest that a new maintenance concept for aircraft may be appropriate for the 1990s.

On-Condition Maintenance Monitoring for Commercial Aircraft

Maintenance of aircraft used by the commercial airlines is regulated by the Federal Aviation Administration. The airline companies may perform preventive maintenance at scheduled intervals or they may conduct preventive maintenance by replacing engine components at specified performance thresholds. Many of the airline companies adopt the second approach as the most cost-effective from the standpoint of maintenance and loss of revenue because of aircraft unavailability. The performance threshold approach requires a system to monitor performance thresholds such as temperature, pressure, and fuel consumption during normal engine operation. This on-condition monitoring capability evolved from the Aircraft Integrated Data System (AIDS) developed primarily for flight safety purposes in the early 1970s. Monitored data were analyzed to identify components requiring maintenance. However, the analysis was conducted off line, which introduced significant delays between recording of the data and the preventive maintenance action to replace components.

In 1979 some of the airlines pioneered in transmitting the on-condition monitoring data to ground facilities for immediate analysis to identify components that should be scheduled for replacement. This approach is being used today by TWA, Delta, and United Airlines for scheduling maintenance on the DC-9, the Super 80, and the 757/767 aircraft. The ARINC Communications Addressing and Reporting System (ACARS), augmented with an auxiliary terminal, is used for transmission of the data (see Figure 1). Typically, engine-monitoring data are transmitted at 5-minute

*Member IEEE.

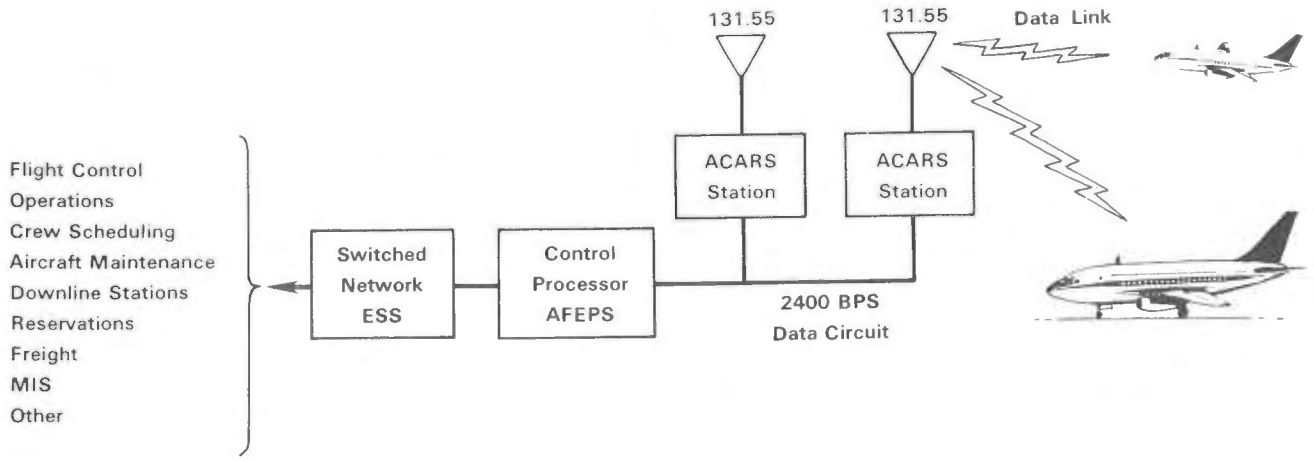


Fig. 1 ARINC Communications Addressing and Reporting System (ACARS)

intervals during a climb and 30-minute intervals during cruise. A single VHF channel (131.55 MHz) is used by ACARS for transmission of digital information. Maintenance-monitoring data represent only 15 percent of the current digital traffic load; however, this will increase as more airlines implement remote maintenance monitoring. Once received at the ground station, these data are analyzed to determine the subsystems requiring replacement. The aircraft flight schedule is then analyzed to determine the appropriate location to accomplish preventive maintenance with a minimum disruption of operations.

Concept for Avionics Interconnected Maintenance System (AIMS)

The airline community is now considering an expansion of its aircraft maintenance concept. Aeronautical Radio, Inc., has proposed a concept for using ACARS to transmit avionics fault information derived from BIT data. The concept, illustrated in Figure 2, is to format BITE (1) results into an ACARS (2) message and transmit the message to ground maintenance control whenever a system failure occurs. Ground maintenance analyzes the data with an "expert system" (3) that emulates the diagnostic logic of the maintenance engineers to diagnose faults. Should additional data be required, query messages are sent to the aircraft until a diagnosis is made. Once the diagnosis of the fault is completed, flight operations (4) is queried for a flight plan and a schedule is established to perform the required maintenance. The appropriate maintenance activity (line station) (5) is notified of scheduled maintenance, stores (supply) (6) is notified of the required component, and the Avionics Test Shop (7) is notified of the fault condition of the LRU being replaced. Finally, a data base file that records the maintenance activity is updated (8).

Military Maintenance Monitoring

The military has pursued on-condition monitoring in parallel with the commercial airlines. The

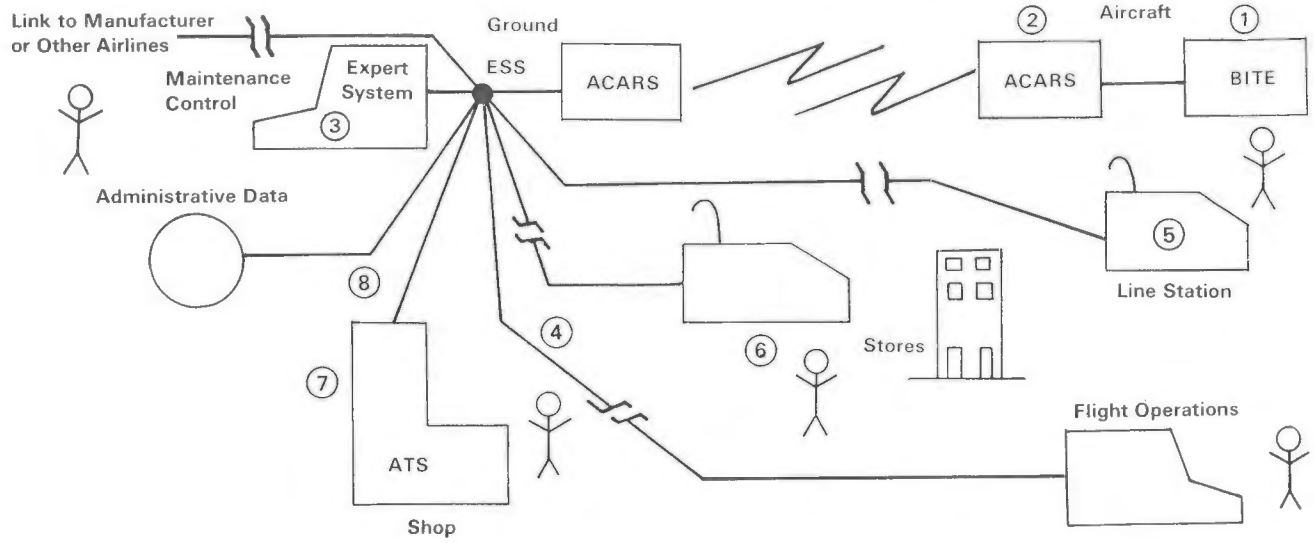


Fig. 2 Airlines Concept for Avionics Interconnected Maintenance System (AIMS)

first effort in on-condition monitoring was with the C-5A aircraft. The Malfunction, Analysis, Detection, and Recording System (MADAR) monitored more than engine performance [1,100 test points, including avionics, engine vibration, pressure, temperature, and airframe stress (1)]. Initially, the monitored data were transmitted to ground stations for processing. Difficulties with workload during missions and the quality of the data forced a modification of the system to replace the direct data transmission with a recorder and off-line processing with a human quality control interface. Over the years this system was used, and an extensive data base of recorded performance information was established.

A more recent effort in on-condition monitoring is the Turbine Engine Monitoring System (TEMS) for the A-10/TF-34 engine. This program consists of in-flight and ground hardware to sense and analyze engine parametric data for fault detection, isolation, and trending. Currently, the Air Force is completing a Squadron Integration Program that will integrate the TEMS capabilities into the maintenance and logistics capabilities for the A-10/TF-34 engine before operational implementation of the system. TEMS is complemented by the Comprehensive Engine Management System (CEMS), which is a ground-based system that supports the engine management community with trend analysis of performance data, engine status, and inventory control. Although TEMS will interface with CEMS, there apparently is no serious consideration being given to transmitting TEMS data to CEMS ground station via a data link. Additional information on the TEMS concept is available in References 2 and 3. The Navy has a comparable program for engine monitoring for the A-7 aircraft, referred to as the Engine Integrated Consolidated Maintenance System (EICMS).

The military implementation of on-condition monitoring presents a different scenario than the airline implementation. Commercial aircraft will normally fly standard routes with minimal change in engine conditions, whereas military aircraft engines are subject to continual change in engine condition. This presents difficulty in establishing a baseline for on-condition monitoring. The military has, however, supported on-condition monitoring as part of reliability-centered maintenance.* To implement this policy there is an Integrated Turbine Engine Monitoring System (ITEMS) being initiated to expand the TEMS concept to new engines under development.

Central Integrated Test System (CITS)

The B-1 aircraft also has a planned in-flight engine performance monitoring system as part of the CITS. The engine monitoring does not encompass the sophistication of TEMS (primarily time and temperature data) but does include maintenance monitoring of avionics subsystems. Failed subsystems are identified with BIT and the information displayed for air crews. Data from CITS are recorded for later analysis using ground maintenance facilities.

*Reliability-centered maintenance is a maintenance concept that allows the condition of the equipment to dictate the need for maintenance or the extent of repair required.

F/A-18 Avionics Fault Tree Analyzer (AFTA)

The Navy's new digital F/A-18 aircraft employs extensive BIT for avionics. Many of the F/A-18 avionics subsystems have gone beyond the requirement to isolate faults to WRAs (Navy equivalent of LRU) and are actually providing data to isolate to SRA (Navy equivalent of SRU). The mission computer is used to integrate the BIT data for all WRAs into fault messages that are maintained in processor memory. A flight-line tester, referred to as the Avionics Fault Tree Analyzer (AFTA), has been developed to access and analyze the BIT data (4). The analyzer serves as an "expert system" that conducts a diagnostic analysis emulating an experienced maintenance technician in order to isolate a fault to the SRA level. Although the AFTA is currently ground support equipment, the functions performed could be incorporated into the airborne processor or in a ground system with fault data transmitted to the ground station using a data link.

A Proposed Maintenance Concept Using a Data Link

Although the concept of using a data link for on-condition engine monitoring has not been adopted within the military to date, there seems to be sufficient motivation for considering this approach for the 1990s. Figure 3 illustrates a concept for remote maintenance monitoring that integrates both engine monitoring and avionics fault analysis with ground processing facilities using a data link. The proposed aircraft system in Figure 3 incorporates the capabilities of the F/A-18 AFTA integrated with the on-board computer. The AFTA currently is a passive device that has been designed to analyze the results of BIT data. To facilitate isolating faults to the SRU level, it may be necessary to augment AFTA with a limited capability to generate test signals. The mission computer would provide the interface between the on-board fault analyzer (AFTA) and a data link to the ground maintenance system. Avionics fault and engine-monitoring data would be formatted into maintenance messages for transmission over a data link. Messages from ground maintenance facilities requesting additional data would be processed by the mission computer, which issues appropriate commands to the fault analyzer.

The concept of operation for providing maintenance status information to ground stations would be to initially transmit baseline information at the beginning of a mission. Subsequent messages would be sent only when the status changes outside preset limits. The volume of data for this type of operation would place minimal demand on a data link system. Should further system design reveal that ground communications facilities would not be within range of aircraft, the system could store data and provide a "dump" of monitored data when aircraft are within range of a ground station.

The ground system centers around the AIS maintenance facility but includes interfaces with both flight operations, specialized repair centers (SRCs), and the supply system. The avionics fault data will be analyzed to identify SRAs needing replacement. Expert systems supported by highly trained technicians at SRCs will query the mission computer for additional diagnostic

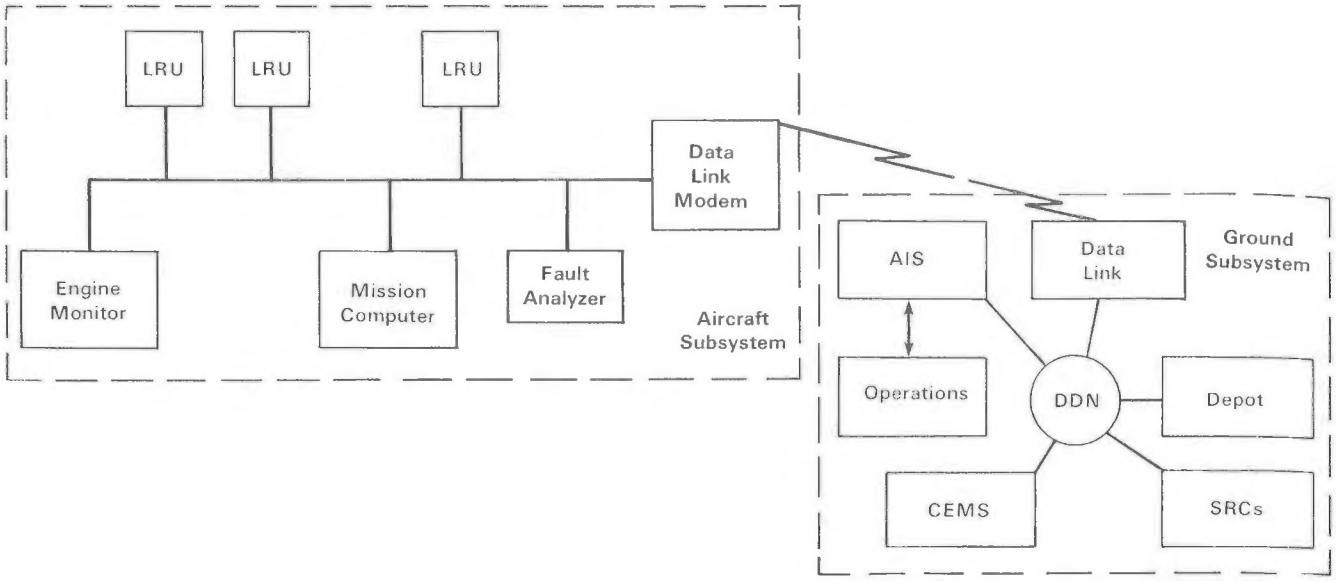


Fig. 3 Proposed Maintenance Concept Using a Data Link

information to resolve ambiguities and isolate faults to the SRA level. By using this approach, diagnosing failures should be more effective than the classical approach of trying to reproduce the maintenance problem in a "shop environment." Once the faulty SRA is identified, the system locates a replacement in the supply system. Immediate action is taken to have the part dispatched to the base where the flight will terminate so that the faulty SRA can be replaced and the avionics restored to full operational capability. The current alternative to this approach is to wait until the aircraft has returned to base and is turned over to maintenance personnel.

Although the time advantage in transmitting fault information over a data link may not seem significant for a short mission, the additional lead time until maintenance personnel are actually diagnosing faults can result in a significant delay. In an environment in which rapid turnaround of aircraft to maintain a high sortie rate is important, the use of a data link to provide the maintenance facility with fault data in real time becomes very significant.

In addition to transmitting avionics fault messages over a data link, engine-monitoring messages would similarly be transmitted using the same facilities. The main difference in the engine data would be the actions required by the ground maintenance system. Engine-monitoring data do not necessarily indicate an existing fault condition and, as such, urgency of maintenance action is not the issue; rather, the issue is scheduling preventive maintenance in an efficient manner to optimize maintenance resources and minimize the unavailability of the aircraft.

Data Link Requirements

As mentioned previously, the volume of maintenance data that would be transmitted by each aircraft for status reporting of fault information is minimal. The volume of data for engine-monitoring messages would be somewhat greater but would still be considered a minimal demand on the data link system. This is presuming that the frequency of transmission of the engine-monitoring data is comparable to that employed over the airline ACARS (between 5- and 30-minute intervals). The data requirements for the system would mainly be determined by the number of aircraft using the system in a given area. Assuming a maximum of 50 aircraft using a single 3 KHz channel with a digital modem similar to ACARS, there should be adequate capacity.

An alternative to using a voice channel would be to establish a series of maintenance messages that could be transmitted over planned data links such as the Joint Tactical Information Distribution System (JTIDS). This alternative would only accommodate aircraft equipped with JTIDS, which, in the case of the Air Force, may be a small percentage of the aircraft inventory. Similarly, the Navy will not have a JTIDS data link capability on many aircraft that will be operational in the 1990s. Although the data link capacity available through JTIDS would accommodate low-priority maintenance messages, the software changes needed to process additional messages could present problems at this stage in the development. For these reasons a UHF voice channel with a digital modem, similar to the ACARS digital modem, would appear to be the preferred approach. The ground communications system to interconnect operators with maintenance and supply facilities would use existing digital transmission facilities such as the Defense Digital Network (DDN).

Advantages of Remote Maintenance Using a Data Link

The concept of using a data link to transmit maintenance data to military ground facilities could have the following significant advantages:

- Avionics faults would be identified to an LRU and to the extent possible, to an SRU,

during a mission. Replacement of faulty LRUs and SRUs could be accomplished on the flight line immediately upon return from mission, with the aircraft returned to complete operational status in minutes rather than hours or days.

- Those faults which cannot be unambiguously identified to LRU/SRU could be further diagnosed by using an expert system. Additional information would be requested by the expert system from the aircraft mission computer concerning the failure. The response would be used to make a diagnosis.

- Intermittent faults could be diagnosed in actual operating conditions rather than being subjected to extensive bench testing to reproduce the fault.

- A centralized ground maintenance facility with an expert system would reduce some of the specialized personnel and equipment requirements of intermediate-level facilities. This would reduce the size and complexity of the AIS/AIMD organization, thereby providing better mobility and decreased operating cost.

Problems

Implementation of the concepts addressed herein will present some problems that must be recognized. These problem areas are as follows:

- The effective implementation of avionics fault monitoring depends on an effective integrated BIT. Although new aircraft will be employing BIT, the capability would not be available on existing operational aircraft. Consequently, the implementation of maintenance monitoring using a data link would initially be limited to new aircraft. As older aircraft undergo major upgrades of avionics or are replaced, they could adopt the data link maintenance concept.

- On-condition monitoring of engine performance is not receiving complete acceptance within the military as a replacement for scheduled maintenance for reasons addressed earlier. The value of using a data link to transmit engine-monitoring data versus recording the data is of questionable benefit if maintenance scheduling action is not taken upon receipt of the data.

- The ground maintenance system needed to support this proposed maintenance concept would need to integrate operations, maintenance, and supply into a common data link system. Currently, these are considered separate functional areas with supporting communications systems unique to the requirements of each area.

- Currently, a data link capability from aircraft to ground sites is not universally available. Aircraft operating in tactical data systems nets have this capability at the present time, but many aircraft rely on voice nets for air-to-ground communications. The addition of a data modem to provide data link capability over existing UHF voice channels would require the use of additional avionics equipment that could present weight and space problems.

- Aircraft processor capability is often a limitation in aircraft improvements. Planning for remote maintenance monitoring should include provision for increasing processing capability in the development of a new aircraft or in major upgrade programs.

Summary

The trend toward digital avionics in new aircraft emphasizing extensive use of BIT suggests changes in the traditional approaches to maintenance. A maintenance concept is proposed that would employ data links to permit the remote monitoring of maintenance data. Advance notice of faults in avionics equipment and degraded performance of engines could be used to more efficiently schedule maintenance actions, including the location and prepositioning of replacement parts. An improvement in operational readiness would result. Some problems are expected in implementing this approach in the near term, but planning should begin now for an evolutionary implementation in the 1990s.

References

(1) U.S. Air Force Technical Order 1C-5A-2-G, "Instruments, C-5A Aircraft," 13 February 1984, Kelly Air Force Base, Texas.

(2) Presentation to 20th Joint Propulsion Conference, AIAA/SAE/ASME, "Evaluation of Benefits of the A-10/TF34 Turbine Engine Monitoring System Squadron Integration Program," AIAA-84-1414, June 11-13, 1984, Cincinnati, Ohio.

(3) Presentation to 17th Joint Propulsion Conference, AIAA/SAE/ASME, "A-10/TF34 Turbine Engine Monitoring System - Evaluation and Implementation," AIAA-81-1447, July 27-29, 1981, Colorado Springs, Colorado.

(4) "Avionics Fault Tree Analyzer," Michael E. Harris, AGARD* Proceedings #343, Paper #20, 18-23 April 1983.

*AGARD is the Advisory Group for Aerospace Research and Development.

84-2705

CRASH-SURVIVABLE FLIGHT DATA RECORDER

Gregory E. Davis *

General Dynamics Corporation
Fort Worth Division
Fort Worth, Texas

ABSTRACT

The Crash Survivable Flight Data Recorder is a multipurpose, solid state, data recording system designed to meet aircraft requirements for fleet management data collecting and incident recording. The physical design of the hardware has been standardized to meet the recording needs of current and future aircraft by adapting to a specific airframe through interface and software program changes. The CSFDR comprises two LRU's - Signal Acquisition Unit (SAU) and Crash Survivable Memory Unit (CSMU). The SAU uses MIL-STD-1750A processor technology, programmed in Jovial J73, to receive, process, compress and store fleet management data into its four-megabit Auxillary Memory Unit. The CSMU provides for 28,000-word storage of incident data. Details on F-16 implementation and data collection requirements are presented with features identified that enhance the adaptability of the system.

INTRODUCTION

The Crash Survivable Flight Data Recorder (CSFDR) requirement in tactical aircraft has been around for years, but the technology necessary for implementation is just now coming of age. In 1979, the USAF commissioned three contractors to independently study the feasibility, configuration, and system requirements for a CSFDR for use on three candidate aircraft - the A-10, F-15, and F-16. A draft specification, drawn on the study conclusions of the three contractors, was completed in May 1982. Control of the concept shifted to a Tri-Service Working Group formed under the jurisdiction of the Joint Services Review Committee to finalize the draft for a two-box CSFDR concept, that is, a signal interface and processing unit and a crash survivable memory unit. The signal interface and data processing unit is designed to be in an avionics bay (centralized to minimize wiring weight and harness impact in the aircraft) while the crash survivable memory unit for storing flight data is removed to a portion of the airframe deemed most likely to survive. Generally, avionics bays are not "crash survivable" portions of an airframe because the rear-mounted engine(s) in current fighter aircraft have enough inertia to crush the equipment in forward fuselage bays.

Interest in the recorder system for the F-16 program grew as such features as a programmable CPU and data collection algorithms for optimizing the data collection procedures to satisfy user needs became more clearly defined and apparently feasible.

* Member AIAA

The need for flexibility in the system design led the F-16 System Project Office to designate a third component for the system - a bulk memory unit to hold a large quantity of non-crash-hardened memory for collecting the fleet management data that is currently collected by the MXU-553A Flight Loads Recorder. The additional memory unit concept was endorsed by the Tri-Service Working Group since it could be added to the system with little impact.

SYSTEM DESCRIPTION

A block diagram of the resulting system configuration is shown in Figure 1. The signal interface and data processing unit, now identified as the Signal Acquisition Unit (SAU), provides the hardware for operating the system and interfacing the aircraft. The Auxillary Memory Unit (AMU), containing four megabits of nonvolatile memory, has been incorporated in the SAU because of space limitations in the F-16. The Crash Survivable Memory Unit (CSMU) is designed to communicate with the SAU and store flight data in its 464 kilobit (58K by 8) memory. Sketches of the units are shown in Figure 2. The CSMU design allows the crash survivable memory to grow to one megabit (128K by 8). Communication with the SAU is conducted over an RS-422A data link because of the inherent simplicity, reliability, and power requirements. The protocol for the data link allows multiple CSMUs or other similar terminals to operate on the same link. Installation locations for this hardware, in keeping with the earlier comments, are depicted in Figure 3. The SAU is in the aft equipment bay, mounted in the location previously occupied by the converter/multiplexer for the MXU-553/A Flight Loads Recorder. The space freed by removing the MXU-553/A recorder allows airborne video tape recorders to be installed in all F-16s. The CSMU mounts in the box-like airframe structure that supports the right horizontal tail actuator assembly.

Standardization was the marching order behind the system design and the citing of specifications for features within.

Military Standards 1553, 1589 (Jovial J73), and 1750 (16-bit CPU) formed the triad for making the SAU compatible with avionic hardware being designed for the F-16C/D and other avionics modernization programs. The MIL-STD-1553 terminal interface meets the requirements of Revision B and can be reprogrammed in the SAU connector to operate "1553 basic" for aircraft so equipped. Efforts are being made to utilize the F9450 single-chip CPU, under development by Fairchild Camera & Space Microprocessor Division for the USAF, as the MIL-STD-1750 processor.

Processing throughput, program memory, and random access memory (RAM) size are specified to be 30 percent greater than required to meet F-16 data

processing needs, which were judged to be the most extensive of the aircraft under consideration, to allow future growth of the system. Programming the SAU operational flight program (OFP) in a higher order language (Ada or JOVIAL J73) was specified to ensure documentation and adaptibility of the OFP. JOVIAL J73 is used because of compiler availability and its use on other systems within the F-16.

Circuit cards in the SAU were packaged in standard electronic module "C" configuration (SEM-C), although this packaging is not employed for the CSMU. Partitioning the circuitry into these replaceable modules makes it easy to reconfigure the system to meet the particular needs of a given system application. The SAU is not designed to provide a cafeteria of inputs for an aircraft program to rummage through. Circuit board real estate, power, thermal dissipation, connector pin availability, and the peculiarity of aircraft analog signals must all be considered when configuring the input section to meet the interface requirements of a peculiar aircraft. Additionally, the OFP must be rewritten or adapted to support the revised interface and data collection requirements.

Environmental requirements for the CSFDR were consolidated from such diverse aircraft as the F-15, F-18, UH-60, P-3C, and the F-16. Both the SAU and CSMU are designed to be hard-mounted to the airframe, so the vibration requirements are extensive. Endurance levels exceed 30 g's and the sinusoidal vibration amplitude alone for gunfiring is 17 g's over half the frequency range. The two units are required to function without cooling air at MIL-SPEC temperatures to an altitude of 70,000 fleet. In addition, they must tolerate a 30-minute exposure to +95°C. Sand, salt spray, fungus, explosive atmosphere, shock, and acceleration complete the list of required environmental tests. Electromagnetic compatibility requirements add to the design effort because of the high field levels specified. For example, Radiation Susceptibility (RS03) levels for the SAU are 100 volts per meter over much of the frequency band, and the CSMU must meet 200 volts per meter. The result will be a system that can operate in environments much more hostile than would have been considered possible for many current-generation avionics and recorder systems.

The crash-survivability requirements for the CSMU are in addition to those of the operational environment. The charter of the system is that the CSMU retain the most recent twenty minutes of aircraft operating data at all times and specifically after an aircraft crash. The Tri-Service Working Group followed the conclusions of the Aeronautical Systems Division study contractors and set survivability requirements equal to or tougher than the FAA TSO-51A requirements for crash survivable recorder systems on commercial aircraft. Specifically, high-g shock was raised to 1700 g's from 1000 g's to withstand the higher shock loads attained in a fighter aircraft. The 500-lb pin drop and 5000-lb crush tests of TSO-51A were not modified. The 30-minute burn test at 1100°C was retained because the P-3C can carry enough fuel to sustain a 30-minute fire, even though a typical fighter aircraft does not.

Examination of memory devises pointed to the use of NMOS floating-gate EEPROM as the memory medium for the CSMU. A goal of this system is to use

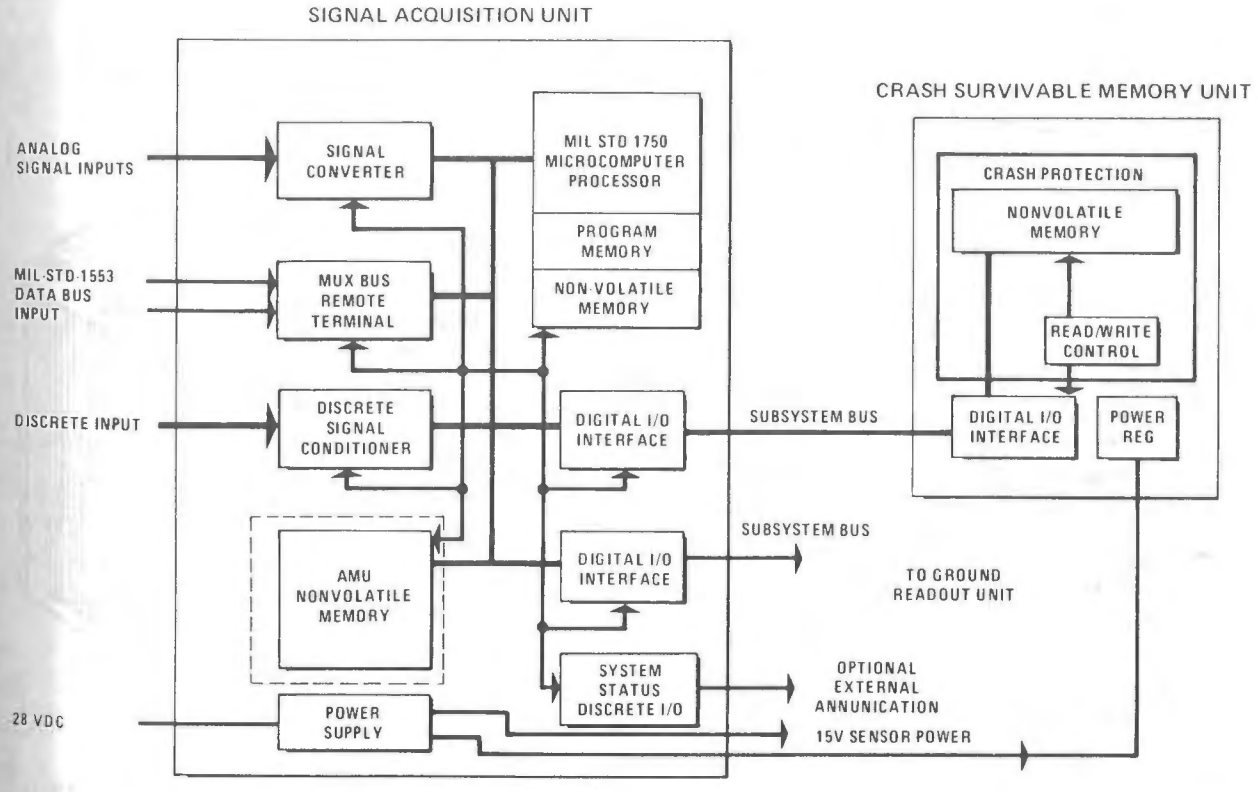


Figure 1. Crash-Survivable Flight Data Recorder Block Diagram

a solid-state memory medium for storage of mishap data. Bubble memory was ruled out because of thermal and shock limitations. CMOS RAM with battery backup would introduce battery problems along with risk of single-point failures killing whole sections of memory. SNOS and MNOS BORAM technologies tend to be proprietary, without second sources, and are subject to data loss in proportion to length of exposure at high temperature.

Mechanical tape drives were not specifically ruled out for the AMU medium, but other specification requirements (reliability, vibration, etc.) drove the memory medium selection to solid state and ultimately to MNOS EEPROMs. Efficiency of packaging space dictated that the AMU consist of one or two cards inside the SAU, and that the cards be removable for aircraft/system applications that do not require bulk storage. The card slots could then be put to other uses.

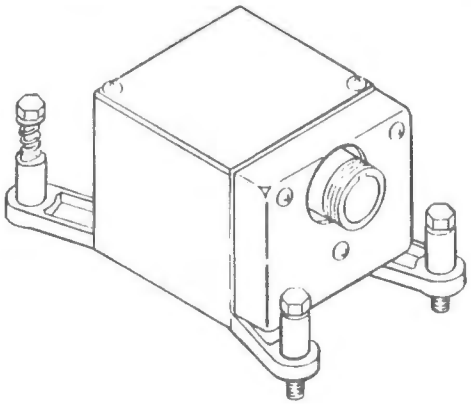
F-16 IMPLEMENTATION

The F-16 airborne data collection requirements extend beyond the need to collect mishap data (referred to as Type 1 data) within the CSMU. Four types of data are collected overall. Type 2 is Individual Aircraft Tracking (IAT) data that provides a matrix of registers for storing peak (positive and negative) g versus the product of peak g and aircraft gross weight. The IAT data allows the individual aircraft to be tracked relative to composite fleet utilization so that high-use aircraft may be scheduled for overhaul before aircraft life is exceeded. Type 2 data requires approximately 1000 bits and is stored in the SAU nonvolative memory section. Type 3 data is Structural Loads Data for collecting aircraft wing bending moments and other factors that determine aircraft service life consumption. Certain parameters are used as triggers to initiate recording of all Type 3 parameters whenever one of the trigger parameters reaches a discernable positive or negative peak. Type 4 data pertains to engine usage, wherein engine parameters and aircraft platform movements are recorded.

Data Types 3 and 4 share several parameters and are recorded together in the AMU. Depending on the dynamics of each flight, approximately ten flights may be stored in the AMU at any one time. Incoming data overwrites the oldest stored data to maintain the freshness of data stored.

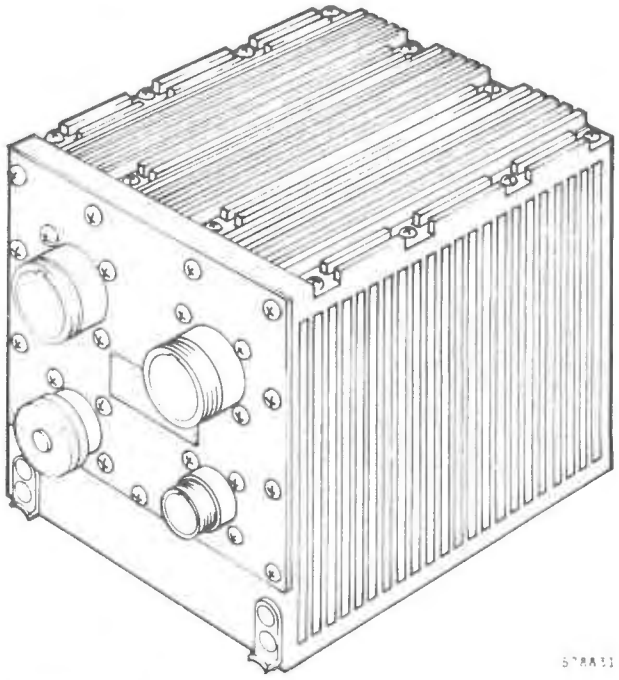
Since structural and engine loads engineers need only a statistical sampling of the entire fleet, the data collection activity has shifted from the flightline to the 100-hour phase inspection. The approximate ten flights of Type 3 and 4 Data in the AMU, along with IAT data collected on a regular basis, will produce a more valid data package for analysis than is currently yielded by existing flight loads recorders.

Current plans call for using the Memory Loader/Verifier (MLV), ASM-660, with a dedicated RS-422 Data Link, to collect the data that is downloaded from the SAU. Choice of the MLV for downloading is not critical. The data link will communicate with any device that uses the DDCMP-



CSMU

- CRASH SURVIVABILITY FOR MILITARY AIR
- WEIGHT – 3.48 LBS
- SIZE – 2.9" x 3.0" x 4.5"
- CRASH SURVIVABLE MEMORY – 28K, GROWTH TO 64K



SAU/AMU

- 1750 MICROPROCESSOR
- PROGRAM MEMORY – 16K, GROWTH TO 24K
- SCRATCHPAD MEMORY – 2K, GROWTH TO 8K
- NONVOLATILE MEMORY – 2K, GROWTH TO 4K
- AUXILIARY MEMORY UNIT – 256K
- INTERFACE – ANALOG
  – DISCRETE
  – 1553
- WEIGHT – 14.35 LBS
- SIZE – 6.2" x 7.0" x 7.25"

Figure 2. Crash-Survivable Flight Data Recorder Components

based protocol and RS-422 Interconnection specified in Standard Interface for Data Transfer Equipment (SIDTE) under development by ASD. Through the keypad on the MLV, the operator specifies the operation to occur. It is possible to access any of the data types or BIT status information in the SAU or CSMU for readout. A Cromenco-based computer station is used to transfer the collected data from the MLV data cassette to DC-300A data cassettes in the intermediate shop for either storage or transmittal offsite. This computer station provides the local user with a means to print out the details of the system BIT status, assess the validity of downloaded data, and decompress/display Type 1 data. BIT status and data validity feedback at the user level allows the maintenance group to receive system status information without waiting for the data to be processed at a remote facility. The decompression and display of Type 1 data allows the local safety or maintenance officers to review any incident recorded by the system. For example, an incident like overstressing the aircraft, where major inspections might be necessary, the CSMU would have the data for determining the intensity and duration of the overstress along with the aircraft weight, altitude, attitude, speed, etc., necessary to determine the severity of the overstress and the inspections needed.

Parameters for the F-16 CSFDR are composed of analog and discrete inputs plus words from the MIL-STD-1553 multiplex bus. Twenty-five analog and 42 discrete signals are used, while only 14 words -- relating primarily to altitude, speed, and attitude -- are used from the multiplex bus. The heavy dependence on analog and discrete inputs is justified by the fact that MIL-STD-1553 buses are dependent on avionic computers to act as bus controllers and they cease to function when the aircraft has lost main and standby generator capability. Parameters must be selected to allow the system inputs to degradate in a controlled manner because the mishap data desirability grows as the functional health of the aircraft worsens. It is important, therefore, for the CSFDR to have access to important analog and discrete signals that relate to aircraft movement and operation. The loss of signals to the CSFDR is trackable relative to time and provides investigators with important clues to assess an aircraft failure.

A second point to be made for careful parameter selection is the amount of data that can be deduced from an initial set of parameters. For example, knowing the pilot input to the control stick and the position of the flight control surfaces, the health of a fly-by-wire flight control system can be deduced.

Many hours were spent in selecting the inputs, sampling rates, data resolution, and ranges for the CFSDR parameter specifications. This is a critical task in successfully integrating the CSFDR into an aircraft. Engineering design groups, aerospace safety personnel, SPO and other USAF representatives, and others met to assemble the list of F-16 parameters to be recorded.

During flight, inputs will be sampled on a scheduled basis rather than recorded on a fixed-frame format. This distinction is important because the data compression algorithms decides what data is to be recorded. The algorithms keeps the data necessary to describe the activity of each input parameter according to the limits of sampling rate, gate width, and

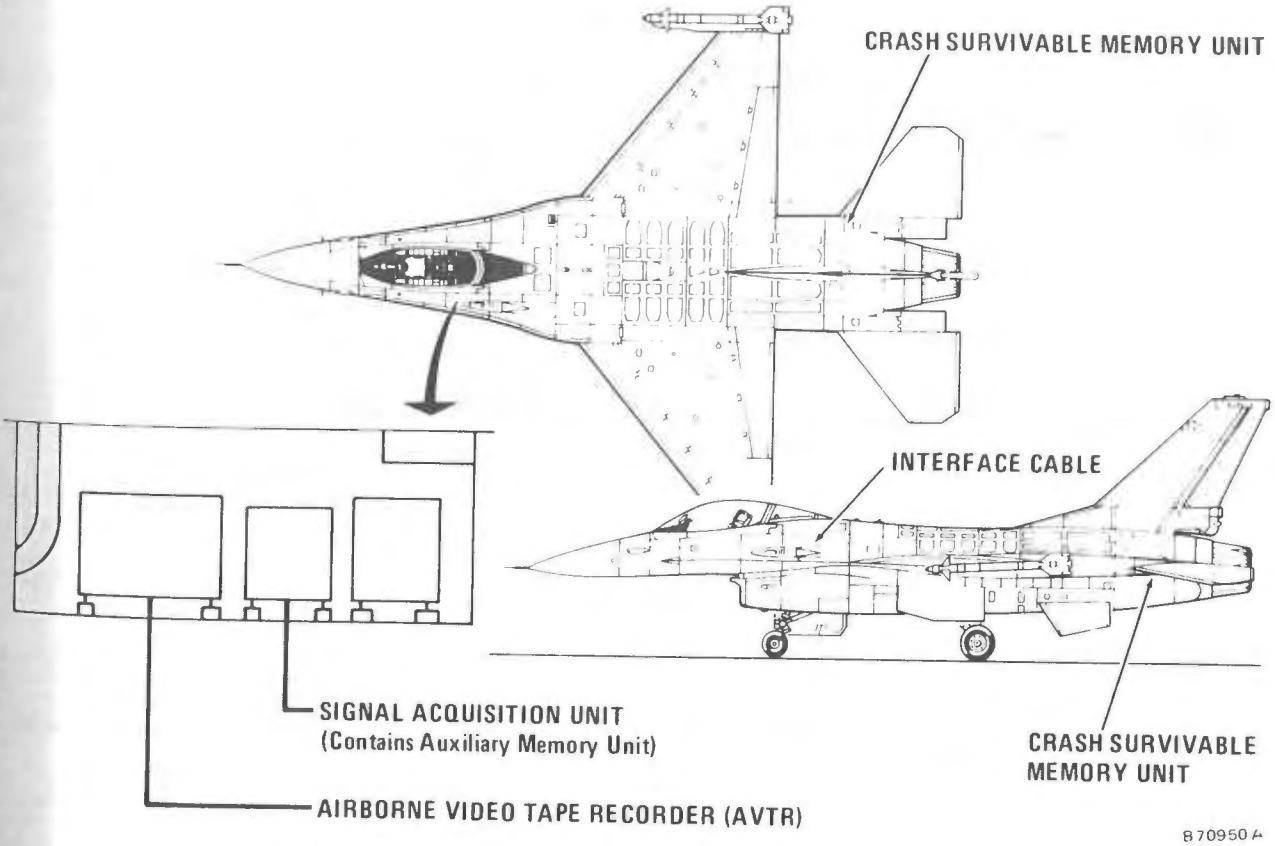


Figure 3. System Installation Within the F-16C/D Aircraft

resolution prescribed. The compressed data is then formed into blocks and tagged with time data and other housekeeping items for transmittal to the appropriate memory for storage. The four data types recorded in the F-16C/D system are conducted on a time sharing basis with the mishap (Type 1) data given the highest priority for processor time. Type 1 data is transmitted to the CSMU on a regular basis to prevent data buildup within the SAU, which isn't crash survivable. One of the primary requirements levied on this system is to convert the necessary signal inputs to digital, sample all inputs, process, select and compress, and transmit the desired data to the CSMU for storage in crash survivable memory -- all within 500 milliseconds.

SYSTEM DEVELOPMENT

The F-16 program was selected to develop the CSFDR in a lead-the-fleet action. The Tri-Service Working Group specification draft and program requirements for standardization (e.g., Reprocurement Data Rights, DAR Clause 7-2003.38 "Notice of Possible Standardization") were passed to General Dynamics which generated the CSFDR detailed specification and procurement package for the F-16 application.

The CSFDR source selection was one of the most intensive for General Dynamics Fort Worth Division. Interest letters were sent to 31 companies, Request-for-Proposals were sent to 19 of the 31, and 12 of the 19 responded with firm, fixed-price proposals. The selected vendor for the CSFDR program is Lear Siegler Instrument Division of Grand Rapids, Michigan. Lear Siegler began development work soon after program award.

SUMMARY

The Crash-Survivable Flight Data Recorder under development for the F-16 represents a new generation of airborne flight data recorders, powerful yet adaptable enough to meet airborne data collection requirements for current and future aircraft. This system is likely to become a required component in future aircraft design projects.

SESSION 18 VLSI DESIGN AND TESTING

Chairmen:

Chao H. Huang
Lockheed R&D Division

This session presents several of the technical issues relating to the design and testing of VLSI chips, with particular emphasis on the integration of design for testability into the VLSI design cycle.

SEMICUSTOM DEVICES AND DESIGN

Chuck Stern

Field Application Consultant

Silicon Compilers Incorporated

Los Gatos, California

The paper focuses on new semicustom components and the CAD tools associated with them that will be used for implementing user-defined integrated circuits in the 1985-88 time frame.

The discussion covers gate array and standard cell methodologies and includes the use of ROM, RAM, PLA and microprocessor cells in each. Software and hardware techniques that allow the user to migrate between gate arrays (traditionally, a fast-turnaround implementation) and standard cells (a low-cost implementation) is covered.

Design examples will be included to graphically demonstrate the current state of such hardware and software tools.

More than ever systems manufacturers are being required to supply more sophisticated equipment in smaller packages, forcing the manufacturers to use VLSI integrated circuits. The reasons for this covers a broad spectrum of applications, running from military (typically low volume) to commercial (typically high volumes). Some of the reasons are:

- Standard components must appeal to a broad variety of applications rather than a specific application; hence, they are, at best, compromises.
- Highly integrated ICs become more function - specific and therefore have a narrower support base. It is a phenomena that the number af transistors/IC has actually decreased on the average over the last few years.
- Rapid changes in manufacturing technology make it hard for a supplier to keep up. Examples: 8048 in nMOS vs. 8048 in CMOS vs. 8048 in ECL.
- Cost associated with using ICs are more a function of the number of devices rather than the type (cost) of the device. Fewer devices mean less cost to the manufacturer.

VLSI Benefits

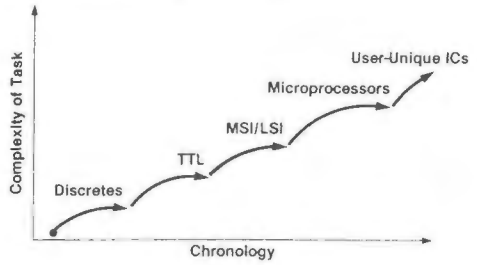
- Smaller Size
- Lower Cost
- Higher Performance
- Higher Functionality
- Higher Reliability

Standard Products Are Limited

- Force Fit for Applications
- Do Not Offer Product Uniqueness
- Difficult for Vendors to Define Sufficient Functionality and Generality
- Best for Memories and Microprocessors
- Insufficient for Optimum System Integration

For the past 15 years or so the microprocessor was used to achieve these goals.

System Design Revolution



But what we've seen over this same period is:

- a shift of emphasis in design cost - it now takes longer and is more expensive to implement/correct/change software than hardware;
- a fairly rapid change from 4 bit up's to 8, 16 and now 32 bit units - all have as a minimum an enhanced instruction set and some completely different instructions sets which mean new software development efforts;
- new groups of generic peripheral devices to support the uP changes - many times not available in a reasonable time span with the introduction of the uP; and
- a proliferation of new high-level languages to support the software engineer - Pascal, Basic, ADA and C.

Microprocessors are then limited in their ability to offer significant advantages over competitive designs.

it is at this point where semi-custom and custom integrated circuits -application specific IC's (ASIC's) - provide capabilities and major advantages over standard logic implementation methods.

User Unique ICs...

The Next Systems Revolution

- Required for a Competitive Edge
- Silicon Density Allows Truly Integrated Systems
- All of the Benefits of Very Large Scale Integration
- Foundries Make Production Available

Two methodologies which we shall discuss here are standard cell and gate arrays.

Gate arrays consist of a pre-defined pattern group of multiple-input NAND (or NOR) gates with unconnected inputs and outputs. Current technologies include ECL, CML, TTL, ISL, STL, and CMOS. At the chip interface level, pins may be configured as inputs, outputs or bidirectional/tri-state drivers. The user, or designer, specifies the interconnect of the gates only, making all logic functions (decoders, multiplexers, flip-flops, register, etc.) from these gates. Most gate array manufacturers support pre-defined "hard" macros and "soft" macros in a hierarchical design system. Soft macros are collections of gates with interconnection already defined which can be used by the designer as if it were a single element. Soft macros can be created both by the gate array manufactures and the user or designer.

In addition to pre-defined interconnect, hard macros also have pre-defined layout, thus permitting timing information on these blocks. Typically, gate arrays are cheaper, faster and less costly to implement than a standard cell design, but are more costly to produce in the long run as they are silicon inefficient. Ability to interconnect gates is very dependent on the routing channel structure and the type of logic (random, regular or bus-oriented) that the designer wishes to use. Ranges of 60% to 90% utilization are common at this time with the majority running between 70% and 80% of the gates used. To achieve maximum routability, some manufactures have gone to variable pitch metal or, more commonly, wider channels in the center of the array and narrower channels along the periphery. Fortunately, these techniques are transparent to the user. PLA's are now just starting to be introduced on gate arrays, primarily because they: 1) end up taking up a fixed area of the array regardless of how efficiently they are programmed by the user; and 2) the fusible-link technology did not lead itself easily to gate arrays or or separate mask areas which could easily be changed. Memory, in the form of ROM, is still not easily implemented and pseudo-RAM is very costly in terms of silicon areas on an array. Hence, applications involving PLA's, ROM, RAM and associated circuits (microprocessor elements) have been implemented using standard cells.

Standard cell technologies currently span ECL, CML, TTL and CMOS. Standard cell methodology consists of the designer using a "catalog" of pre-characterized and pre-defined elements, called cells, similar to the "hard" macros in gate arrays. These cells are then placed within designated areas of the die. Separating these areas are routing channels to interconnect the cells.

This methodology provides for fast design time, but since more than the interconnect is specified by the designer more masks must be made and the longer, costlier and riskier is the fabrication of the part. However, standard cells are silicon efficient as compared to gate arrays and run upwards of 85% utilization regularly. This technique also allows for PLA, ROM, RAM and microprocessor structures.

Beginning in 1985, improvements in CAD tools will allow designers to begin to address the turn-around time efficient/silicon efficient dichotomy that has existed up to now.

Several major semi-custom manufacturers will be offering to take an existing gate array design done on their system and create a standard cell or full custom version of that same part in order to take a part designed with a design-efficient method (gate array) and "map" it into a silicon-efficient method (standard cell/full custom). Thus, a user may expect fast turnaround of prototypes and low volumes of production parts and migrate to lower-cost parts for volume production with the "safety net" of known good gate arrays at a later time.

This can be an almost ideal situation provided, of course, that manufacturing technologies are compatible and simulation/timing analysis models are transportable between the two methods. Most viable vendors will provide for this in the coming years by either making simulations run at the transistor levels (very time consuming but inherently compatible) or by providing translation programs (reasonably fast but potentially inaccurate). The problem is, or course, more efficient use of silicon could be achieved by making use of those functions than standard cells can use (ROM, RAM, PLA, microprocessors) that don't exist in gate arrays.

ROMs, PLAs and microprocessors will all require some sort of microcode assembler, similar to PALASM, in order to be quickly defined. Up to now, most of the PLA programs have allowed definition based on input conditions. New versions must allow definitions based on output conditions as well, as demonstrated by the decimal-to-seven segment decoder PLA sheet show below:

```
9876543210 abcdefg
LLLLLLLLLH HHHHHHL
LLLLLLLLHL LHHLLLL
LLLLLLLHLL HHLHHLH
LLLLLLHLLL HHHHLLH
LLLLLHLLLL LHHLLHH
LLLLHLLLLL HLHHLHH        10 Minterms Based
LLLHLLLLLL HLHHHHH          on 0-9 inputs
LLHLLLLLLL HHHLLLL
LHLLLLLLLL HHHHHHH
HLLLLLLLLL HHHLLHH

9876543210 abcdefg
-----L--L- H......
---LL----- .H.....
-------L-- ..H....       7 minterms defined
L-L--L--L- ...H...          by 7 outputs
--L---LLL- ....H..
--L-----LL .....H
```

Additionally, logic reduction algorithms need to also be supplied to minimize the use of PLA minterms. An exceptionally good PLA generator would make provisions for overlapping minterms to provide a "glitchless" PLA as well. The two logical extensions of this is: a) programs will be necessary for "registered PLAs" - i.e., state machines - reduction and b) EPROMs and EPLAs for users who wish to define microcode or logic interfaces at a later time or wish to correct existing errors. Such technologies would allow a user-unique IC to be a peripheral for both Intel and Motorola microprocessors, for example.

As devices get larger, from the current maximum of approximately 13,000 2-input NAND gate equivalents to 20,000 gates, logic simulators will of necessity, have to become more sophisticated and faster.

Recently, gate array and workstation manufacturers have added high-speed hardware logic simulation capabiliteis to their system resulting in a 100 x speed up (or greater). This capability seems adequate up through 1.25 micron CMOS arrays (20,000 gates) which should be in production by the end of 1986.

The toughest problem to overcome with increasing gate complexities will be automatic test vector generation. With 20,000 gates and upwards of 240 pins the number of test vectors needed to verify operation will increase dramatically. At the present time, most gate array/standard cell manufacturers allow approximately 20,000 Sentry test vectors for production test tapes. This means that testers will have to speed up for two reasons:

1.25 Micron CMOS will clock at 75 MHz so testers will have to speed up merely to run real time tests.

Test vectors will increase and to minimize absolute test time (which is reflected as piece part price) the testers must be capable of running as fast as parts will allow - scaled testing will not be accurate.

Simply put, only the absolutely fastest testers currently available (100 MHz) will be capable of tracking technology for the next five years.

Concurrently, more research must be done to produce the maximum testing of ICs with the minimum number of test vectors, and this test vector generation must be reasonably automated. Current standards of fault grading of test vectors will generate far too many vectors to make economical ICs in the future.

An additional component of testing is the techniques used to implement testing of the design on-chip. Level sensitive scan design techniques are probably the best methods for the next several years. It

is imperative that these techniques be learned and implemented by the design community as quickly as possible. Several of the larger semi-custom integrated circuit companies will be adding built-in testing (BIT) schemes in the 1985-1986 time frame. With the larger densities available then and through at least 1990 this technique will probably dominate the market.

There are several related issues regarding simulation that need to be addressed at this time. Only a few suppliers of user-unique ICs allow simulation above chip level - that is, they don't allow modeling and simulation of a system of ICs. As some multi-chip designers have already discovered this capability is a necessity. As illogical as it may seem, chips developed independently that need to work together probably will not unless they are simulated as a system.

In discussing multi-chip simulations it should also be realized that not all arrays or cells will be completed simultaneously. As a result, the system cannot be simulated (and individual ICs corrected or improved) untill the last chip is designed. Clearly this poses a potential slow-down in the design cycle. The latest trend in system-level simulators allows users to enter "dummy" chips. These "dummy" chips can have their operation described in a high-level programming language so as to emulate the yet-to-be-completed integrated circuit. As a result, a system model can be entered and as individual ICs are completed they can be "plugged in " and run to see if it "plays" with the other chips. This is, of course, an iterative process with accuracy of simulation closely tracking the percentage of completed components in the system. Unfortunately, these simulators are not offered by the user-unique IC manufacturers at the current time but by outside test/measurement companies. This poses problems for the users in terms of transferring models from his IC manufacturers system to a more powerful simulator with inherently different models. If is expected that over the next 5 years that these services of multi-chip simulations with software modeling of uncompleted ICs will be offered by all the major gate array/standard cell vendors.

An additional note: since standard cells can currently supported linear functions any practical simulator for these types of ICs must support these linear modeling tools as well.

Barriers to Integration

- VLSI Design Requires
  - Skilled IC Design Craftsmen
  - Long Lead Times
  - Large Development Costs

OR

- Compromise Design Approach
  - Limited Density
  - Limited Tools

In summary, then, we have seen that gate array and standard cell design methodologies will begin to merge over the next 5 years with a fairly easy migration path between gate arrays (fast turnaround) and standard cells (low cost) at least when both are offered by the same manufacturer. Such devices such as ROM, RAM, PLAs and core microprocessors will probably still remain primarily in the domain of standard cells, if only because they are more silicon efficient in that methodology. Computer-aided design tools for generation of ROM/PLA/state machine coding from input equations is a reality now; improvements will provide for reduction/minimization as well. Smaller device sizes will allow for higher speed and higher gate count devices, but this may be offset by larger packages to accommodate I/Os and more expensive testers to exercise these high-speed/high gate-count parts. Test programs will become longer and hence more expensive to run. Research into new test methodologies other than fault grading must be done and these new techniques coupled to automatic generation of test vectors via the design system. Finally, designers of user-unique ICs will need to incorporate hardware test-reduction methods such as level-sensitive scan design and built-in testing.

GATE ARRAY, STANDARD CELL, AND FULLY CUSTOM: BUILDING BLOCKS FOR THE DIGITAL AVIONICS DESIGNER

Henry L. Owen, III
Michael T. Kopp

Georgia Institute of Technology, Atlanta, GA 30332

Introduction

Very Large Scale Integrated (VLSI) Circuit design options that a digital avionics designer has at his disposal include Gate Array, Standard Cell, and Fully Custom. Each of these options has advantages and disadvantages which must be considered carefully during the design phase of any advanced digital avionics system. The design engineer must understand the capabilities and limitations of the design to be implemented using these VLSI technologies, and the designer must also understand the impact of these technologies on the design process itself.

As VLSI technologies are incorporated into avionics designs, new problems are posed for the design engineer. VLSI allows the design to incorporate a much larger digital network than was possible using more traditional technologies. VLSI, by its sheer magnitude, requires the use of a different design approach because a large network implies more complexity and, thus, Computer Aided Design (CAD) tools are required to make the design of such complex networks feasible.

VLSI design also mandates incorporating "design for testability". Since the digital network contained in a single integrated circuit can be comprised of hundreds of separable functions, careful consideration must be given to testability during the design phase. Testability is included to determine that a good integrated circuit has been fabricated, and to determine if an integrated circuit in place in the system is operating as required.

An additional problem of VLSI designs is that of packaging an integrated circuit that contains more than 100 input and output connections. This large number of connections requires new packaging techniques.

This paper examines the VLSI design options of Gate Array, Standard Cell, and Fully Custom, and contrasts these options along with digital hybrids and programmable logic. The design process for each of the options is illustrated and design methodologies are outlined. Design considerations required in order to use CAD tools in the design process are examined along with testability and packaging considerations.

Gate Array

Many discrete component integrated circuit designs cannot be implemented due to the number of component packages in the system, the power supply sizes required, and/or deteriorating performance of the system with large numbers of signal interconnects. Gate Arrays often provide a cost effective solution to this class of design problems since large amounts of circuitry can be placed inside one VLSI device, without requiring the complex skills and time that traditional Fully Custom integrated circuit designs require. Gate Arrays have been in existence for over ten years, and early applications typically were for computing structures which did not justify the expense or time required to develop Fully Custom integrated circuits.

Gate Arrays capitalize on previously designed logic functions which can be implemented using a partially preprocessed integrated circuit. These integrated circuits consist of components (i.e., resistors, capacitors, transistors) grouped together in units called cells. The cells are made up of a fixed number of components, each of which is available to be used or bypassed. Manufacturers of Gate Arrays have determined the optimal number of components to place in these cells, considering such parameters as the area required for interconnections and the type logic being employed. This number varies from manufacturer to manufacturer due to process dependent parameters. As an example, the optimal number of transistors in a Complementary Metal on Oxide (CMOS) implementation differs from that required in Emitter Coupled Logic (ECL) implementation.

The Cells in a Gate Array are placed on the integrated circuit in rows which are separated by fixed width channels. The channels are initially left uncommitted so that a router can use this area to interconnect the components. The designer personalizes the integrated circuit in the manner required to implement his logic functions. This personalization is typically achieved on the top one to three layers of the integrated circuit. The material used to make the interconnections is usually a metal or polysilicon layer. The remaining layers of the Gate Array contain the non-personalized cells which do not vary from one design to another. This characteristic of Gate Arrays allows a manufacturer to produce these devices in production quantities and still make available a large number of uncommitted gates which are readily available for personalization. This allows the foundry costs of the common portions of the Gate Arrays to be divided among a large number of customers.

The digital avionics designer's task is to determine the pattern required on the top one to three layers of the Gate Array to implement the specific design function. In a Gate Array, the cells are "tried and true," and do not (in theory) contain any potential bugs that will be discovered during the application. The most common logic functions that a digital designer desires to use are often predesigned and tested, and available for use by simply calling that function from a macro library data base.

As an example, if the digital designer desires to implement a four-bit binary counter, and a design for one already exists in the macro library, that design can be entered directly into the designer's circuit. This approach allows a "Fully Custom" like integrated circuit design wherein every single component on the chip is designed for a specific system, except that in this case previously designed and tested macros are used. This reduces design time and complexity at the cost of some loss of flexibility and chip area utilization efficiency. Usually not all of the components in any of the cells are usable; thus, some components and chip area are wasted. Furthermore, traditional routing constraints and the resulting increase in design complexity do not allow individual components to be shared among cells.

The characteristic of wasting some area by not being able to use all of the components in a cell is being examined. Two approaches which attempt to increase the utilization of available components include "gate-isolation" and "channel-less" Gate Arrays. Gate isolation arrays use continuous rows of components (transistors) across the integrated circuit. After a logic function is implemented in a given area, the function is electrically isolated by connecting the border transistors in the off mode. "Channel-less" arrays place components over the entire integrated circuit and allow interconnects to pass through its cells. This method requires that cells accommodate wiring in their interiors, often a costly requirement.

The actual interconnection of components on a Gate Array is determined by use of placement and routing algorithms incorporated in Gate Array software design systems, while the digital designer determines the manner in which the macro functions on the Gate Array are to be connected together in order to achieve a system function. The placement algorithm determines where the macros should be placed to minimize the lengths of the interconnections between them, and the router algorithm determines how to make the interconnections internal to each macro and between macros. The placement of the macros and the channel widths used for interconnection are constrained by the predetermined design of the Gate Array. As a result, routers typically require manual assistance to complete all the interconnections. Depending upon circuit size and density, the router may automatically route all connections, or a few connections (typically three percent) may require manual routing.

Gate Arrays are available in many different sizes. The size of Gate Arrays is usually measured in terms of the number of two-input NAND logic gates one could implement based on the number of transistors contained in the Gate Array. This figure of merit does not take into account the components wasted due to inefficiency. A resulting rule of thumb is that only 80% of the possible NAND logic gates are usable. At present, arrays of up to approximately 10,000 gates are available.

The technologies available in Gate Arrays include n-channel metal oxide semiconductors (n-MOS), Complementary Metal Oxide Semiconductor (CMOS), Emitter Coupled Logic (ECL), Transistor-Transistor Logic (TTL), and Current Mode Logic (CML), to name a few. The most widely used of these technologies is CMOS (with silicon gates being used instead of metal gates since silicon gate technology reduces parasitic capacitances and the gate area required).

In late 1983 there were approximately 75 Gate Array vendors in business, attesting to industry's wide acceptance of Gate Arrays to implement VLSI circuit designs.

Standard Cell

Another implementation option for digital avionics systems is Standard Cell technology which is one step closer to a fully custom integrated circuit. In designing a Standard Cell integrated circuit a designer starts with a blank piece of silicon (the size of which is not yet determined) and implements personalized digital logic on all layers (as an example, nine layers) of the integrated circuit. This process is similar to the Fully Custom design approach, except that only previously tested and defined logic functions are available for use at the design implementation stage. Each of these logic functions is stored in a macro based library as in the case of the Gate Array, but the macro for the Standard Cell contains all the information needed to customize every layer of the integrated circuit, not just the top layers. The designer has available to him many previously designed macros, and can use his/her own digital design skills to interconnect them into digital systems that implement the desired overall function. The process of designing the macro logic functions is a small fully custom design service which must be carried out by the manufacturer, not by the application design engineer.

As part of the design of a Standard Cell macro function, the manufacturer must determine the optimal size for each macro. This physical size is influenced by several factors which include the surface space required to interconnect the various circuit elements inside the macro logic function, and the height to allow enough room to place the CMOS transistors and "guard rings" (commonly used to prevent a phenomenon called latch-up in CMOS circuits) in the macro function. What is commonly done is to determine such a height dimension for each macro function. As an example, a CMOS Standard Cell manufacturer may define this height dimension to be 180 microns.

The physical length of the macro function is determined by the number of circuit elements that must be incorporated to implement the given logic function. As an example, a flip-flop would be a short macro (i.e., 160 microns) whereas a four-bit binary counter would be a long macro (i.e., 1680 microns).

In laying out the macro-circuit elements, a manufacturer will usually opt to place all of the input and output connections on both the top and the bottom of the macro. This enhances the router's ability to interconnect the macro

functions in a system since an electrical connection can now be made in one of two places, instead of being constrained to only one place. Another routing enhancement technique is to place feedthrough connections in the macrofunctions so that if a signal needs to get "around" a given macro on the integrated circuit it can do so by going to a feed-through connection on the top, and then exiting via the feed-through connection on the bottom.

In designing the Standard Cell macro functions the manufacturer can design several identical logic functions, but have different power consumptions and drive capabilities. As an example, for a non-inverting buffer which has a small fanout and is electrically connected to only a few other macros, a "standard" non-inverting buffer may be used. If that non-inverting buffer were required to have a large fanout and be electrically connected to a large number of other macros, a "super-buffer" would be used. The "super-buffer" would be physically larger and require more power consumption; however, it would implement the same non-inverting buffer logic function.

In a Standard Cell integrated circuit, after determining what macros are required to implement the system functions, the designer needs to decide on the physical placement of these macros on the silicon. An additional degree of freedom that exists with Standard Cells (but not with Gate Arrays) is that the macros can be placed anywhere on the blank piece of silicon, and all layers can be specified by the macro itself. In practical Standard Cell implementations, the placement of these macro functions is automatically done using an algorithm that attempts to group the macros closest to its input/output electrical connections to reduce the interconnection lengths. The macros that are grouped together are placed on the same row or nearby rows. The channels are free from macros and are used to interconnect the various macros in the rows. These routing channels are not fixed in width as in the case of the Gate Array. The widths are determined by the number of electrical interconnects that must pass through that channel. Thus the channels will vary in width between different rows. Automatic placement of macros is also optimized to yield a square integrated circuit unless packaging requirements dictate a non-square shape. In optimizing for a square integrated circuit, the placement of the macros requires numerous tradeoffs between the number of macros placed on the longest row, and the number of channels separating the rows, versus functional grouping of the macros, and the width of the channels - those widths being a function of the way that the interconnections were placed in that channel.

The use of artificial intelligence for automating integrated circuit placement and routing is a current area of research. In high performance (short propagation delay and large bandwidth) standard cell circuits, the lengths of some interconnections may be unacceptably long because of the additional stray capacitance and resistance. Also, in large CMOS Standard Cells, these long runs may cause the propagation delays for that circuit node to violate acceptable limits. Manual intervention of the automated process may be required to group macro functions which must be placed adjacent to each other in order to obtain the shortest interconnections possible. The use of artificial intelligence possibly could enhance the placement process to a level of sophistication and results that surpass human capabilities.

In Standard Cell placement and routing, all of the macros can be placed in any location desired, and routing channels may be expanded to accommodate any number of interconnections; thus, every connection can be automatically made. This differs from Gate Array routers for which the freedom to place macros and channel widths is constrained by the predefined elements present on the integrated circuit.

Since Standard Cells are one step closer to Fully Custom designs, they have higher logic packing densities than Gate Arrays. In order to implement the same logic functions a Gate Array requires approximately 1.3 times the area of a Standard Cell, because of the unused components and layout inefficiency of Gate Arrays. This size differential is important to consider in terms of circuit performance and production costs. A smaller circuit will allow macro functions to be located closer together. Such placement can enhance circuit performance because shorter routing paths mean less stray capacitance and resistance in the interconnections, and thus less propagation delay. From the production standpoint, smaller integrated circuits cost less to produce; thus, if the end volume of circuits is large, a cost savings in production can be achieved.

In considering the cost differential between Standard Cells and Gate Arrays, it should be noted that the development costs for Standard Cells are greater than for Gate Arrays. In Standard Cell development every layer of the integrated circuit must be designed by the digital designer, and, therefore, a mask that contains the circuit patterns desired must be fabricated for every layer. Gate Arrays need only a few masks to be generated to specify the top one to three layers. This difference in the number of masks which must be generated for a specific integrated circuit translates into a development cost differential between the two methods. The development time required for Standard Cells differs from that required for Gate Arrays for the same reason. Typically two additional weeks are required for the generation of the additional Standard Cell masks, and the additional fabrication steps required.

In late 1983 there were only approximately 25 Standard Cell vendors in the business. The lower number of Standard Cell vendors is due in part to the higher initial investment required to get into the Standard Cell market. In developing Gate Arrays, funding is required to develop a basic Gate Array layout, to develop macros, and to help the user in the logic implementations. In developing Standard Cells, a much larger investment is required to develop a complete Standard Cell macro library.

Fully Custom

In the Gate Array and Standard Cell options, a library of previously designed Fully Custom macros is available to construct the circuits. These Fully Custom macros were previously laid out and tested, relieving the digital designer from carrying out that task. In designing a Fully Custom integrated circuit, the designer starts with a set of design rules, and then constructs the integrated circuit element by element. The design rules are relatively complicated in that many different rules must be considered during the construction of every portion of the circuit. These complexities, and the detailed nature of the procedure to design a Fully Custom integrated circuit require that the designer be an integrated circuit designer as well as a digital designer.

The Fully Custom option allows the designer the largest amount of freedom in placing his circuit on an integrated circuit. This additional flexibility results in the highest possible density of components on an integrated circuit because the components are placed one by one in an optimal manner. However, in comparison with other design options, the Fully Custom option is the most expensive, requires the longest design cycle, and has the lowest probability of first time success. The cost and complexity of using this design option has traditionally restricted the application of this approach to circuits which have very high production volumes, or must be as small as possible in order to meet design specifications.

The Gate Array and Standard Cell design options evolved from the Fully Custom option and allow the designer to use tested results of previous designs for portions of his circuit. They are cost effective approaches to the design of an application-specific integrated circuit and reduce the time and cost from those associated with Fully Custom integrated circuit designs.

Digital Hybrid

The design philosophy for digital hybrids is different from the design philosophy for integrated circuits in that hybrids are used to increase the number of integrated circuits per unit area rather than to increase the number of logic functions per integrated circuit.

In digital designs where traditional small scale integration (SSI) and medium scale integration (MSI) packages are used, a large proportion of the circuit card is consumed by the packages, since the integrated circuit chips themselves are much smaller than their packages. For hybrid designs, several unpackaged integrated circuit dies are placed on a substrate and connected to obtain the desired circuit configuration. This substrate and its interconnected dies are placed into one larger package. Since integrated circuit dies which perform the common logic functions are relatively easily obtained, prototypes of digital hybrids can be developed inexpensively.

This approach to implementing a design is included in this paper for perspective, despite the fact that it is not usually considered an application specific option because a personalized silicon die is not created.

Programmable Logic

When the projected development time and costs indicate that other options are not cost effective, the use of Programmable Logic devices may be justified. Traditionally, only five to ten Small Scale Integrated (SSI) and Medium Scale Integrated (MSI) circuits could be combined and placed into a single Programmable Logic Device. Recent advances have dramatically increased the capacity of a single Programmable Logic Device to the point where, now, as many as 5,000 gates are available.

The design methodology for these types of devices remains the same. Adjacent packages of SSI and MSI functions are grouped into one module. The input and output pin requirements of the module are specified. Logic equations for the module are tailored to reduce the input/output product-terms by using such logic minimization techniques as Karnough maps. Once the equations have been determined, a programming module is used to blow fuse patterns contained in the "blank" Programmable Logic Devices shipped from the factory.

An early example of a device of this type is the Programmable Read Only Memory (PROM). A more recent example is the Field Programmable Logic Array.

Devices of this type are efficiently used when short turn-around times and fielded system modifications are anticipated.

Design Process and Methodology

The process involved in designing VLSI integrated circuits is actually the process of designing a system. The building blocks used to implement advanced digital designs are sufficiently complicated that they can be considered systems in and of themselves. The method by which these systems are efficiently designed must employ Computer Aided Design (CAD) tools which are used in conjunction with a structured design approach.

Integrated circuit designs have historically been carried out by specialists who were very good at some subset of the tasks required to complete the chip design. One engineer would perform system specifications, another would carry out detailed design, and another would perform the layout of the chip. This approach has been dramatically changed by CAD tools which allow one engineer to carry out all the tasks required to design the chip. This is particularly true in the Gate Array and Standard Cell approaches since previously designed and tested functions are combined in different ways to accomplish various system specifications.

In carrying out a structured design approach, the designer may use CAD tools at all levels of the design process. Initially, a

system specification document must be generated, which includes system level requirements for the design. This system specification is also translated into functional models which are used to evaluate the system's performance. These models are used in later design stages to compare detailed design functionality with the system specification. The functional models do not incorporate detailed timing information, since the detailed design has not been initiated and such information can be derived only from the finished detailed design. The functional models do, however, incorporate the interface timing requirements so that circuit modules will operate correctly with each other.

The next level in the structured design phase is that of system architectural design and tradeoff evaluation. Various architectural tradeoffs are evaluated to determine the optimal approach to achieve system specifications. CAD tools may be used to generate models for all of the approaches being considered, and performance comparisons can be made. Potential architectural bottlenecks in the system can be identified and more closely evaluated during this stage of the design process.

Once the architecture has been determined, the system is broken down into modules. These modules are designed separately and must perform within the specifications set for each module. In a structured design, rules are established for each hardware module so that independent designs may be carried out efficiently and then be interconnected to the system model. During the design process, portions of the system model may consist of detailed designed circuit modules while other portions of the model may still be high level functional models. This flexibility allows the designer to work on each portion of the circuitry in an efficient manner.

Simulators model the system performance at various levels of detail depending upon the stage of the design process the system is in. A simulator used at the very beginning of the design process would be a functional simulator and would not require much detail. As the design of modules moved to detailed logic designs, a logic simulation would be used to evaluate the logic states of the design as a function of the inputs. The most detailed level of a simulator used in the structured design is that of a circuit simulator which models devices at the level of detail of mathematical models of the circuit devices. The effects on circuit performance of the stray capacitance and resistance of interconnecting runs can also be taken into account in the circuit simulator. This level of detail is so great that modeling an entire integrated circuit at this level is not practical in most cases due to the computing power required.

When the detailed design has been completed for a Gate Array or Standard Cell, the design is entered into the CAD system in a format that specifies the circuit elements and their interconnections. After this has been entered into the design system, functional simulations are carried out to verify that the circuit, when fabricated, can perform as designed. After the logic design has been completed, the most time-consuming task involved in the implementation of a Gate Array or Standard Cell is the detailed simulation. These simulations must completely characterize the integrated circuit in such a manner that the simulation results may be used as the specifications for the final product. The simulations result in a set of inputs and outputs that completely characterize the functionality and timing associated with the design. Design engineers are required to invest significant effort in developing these input and output characterization patterns since, being the designers of the circuit, they are the ones best suited for evaluating the results of such simulations.

A simulator which is commonly used by many manufacturers of Gate Arrays and Standard Cells is the "Test Vector Generator and Simulator (TEGAS)." Each manufacturer's application engineers develop a library of their logic macro functions, and place them in a format that the digital design engineer can use to model his particular interconnection of these macros in his digital system. The digital design engineer then specifies a complete set of inputs versus time and the simulator models his circuit and produces the outputs at the time they are expected from the actual integrated circuit. This allows a detailed analysis of the actual design so that performance can be evaluated prior to fabricating the circuit. Additionally, these inputs and predicted outputs are used in the manufacturing process to stimulate produced chips both before and after packaging to determine if a given chip functions as specified and can be placed in the operating system.

In the Gate Array and Standard Cell design process the next step after simulation is to place and route the circuit. Placement is usually done automatically by CAD tools and results in optimal placement of logic functions on the silicon wafer. Routing is also automatically done and results in the various logic functions being interconnected in the prescribed manner to implement the system as designed.

Simulation is required again, after the placement and routing functions have been completed, to evaluate the impact of the placement and routing on the operating speeds of the circuit. In Complementary Metal on Silicon (CMOS) Gate Arrays and Standard Cells, long interconnection lengths between logic functions result in stray capacitances and resistances that can slow the logic transition times to such an extent that the circuit may no longer meet specifications. A successful pass of the circuit through the simulator with these routing parameters taken into account gives strong assurance that the actual device will operate acceptably.

Fully Custom design methodologies are more mature than Gate Array and Standard Cell procedures, and also require more integrated circuit design expertise. Gate Array and Standard Cell design methodologies are simplifications of the Fully Custom design procedure. As such, they

do not require a true integrated circuit designer, but instead can be completed by a digital designer. The Fully Custom option allows the designer the most freedom in specifying what is placed on the integrated circuit, and along with this freedom go design rule constraints which are more complicated than those for Gate Arrays or Standard Cells. The designer of a Fully Custom integrated circuit is required to work at a very detailed level of integrated circuit design and must therefore apply skills that many digital designers do not possess.

The design methodologies for Digital Hybrids and Programmable logic devices parallel more traditional digital design techniques. The most common approach is to divide a previously designed digital circuit into packageable functions and then place them in either a Hybrid or a Programmable Logic Device.

Testability

When digital avionic systems are designed using the building blocks outlined in this paper, each of the resulting integrated circuits is sufficiently complex that it may be considered a small system in itself. So complex a circuit cannot readily be adequately tested in either the manufacturing process (where chips are tested both before and after packaging) or in the fielded system (where go/no-go tests are performed) unless adequate provision for testing has been made within the integrated circuit. The only way to avoid untestable systems is to design the testability into the system itself from the very beginning. A hardware penalty will be incurred by designing the testability into the circuit, since the goal of making circuit nodes both controllable and observable (a fundamental principle of testability) can be achieved only by providing additional circuitry.

The designer must be concerned with testability from the start of a design, and must make the hardware testability tradeoffs required to achieve a produceable design. In a microprocessor environment these tradeoffs might require that the microprocessor input/output bus be internally connected to portions of the circuitry that do not functionally require set-up or control, but are operated in a different mode for test purposes. This additional circuitry might, for example, break into the normal data path in the device by multiplexing in known test data patterns, and allow outputs to be observed through the same I/O bus structure, thus achieving controllability and observability at critical points in the circuit.

Such additional circuitry that is compatible with the input/output bus provides the capability to test a circuit before or after packaging during the manufacturing process, as well as in the fielded system. This particular approach in its simplest form does not necessarily permit functional tests to be performed at operating speed, since a microprocessor input/output write cycle must be used to supply the test data, and this cycle might be longer than the actual input data cycle to the device under test.

Another testability method that is often used employs additional input/output connections on the die that are used only during manufacturing tests. Thereafter the connections are not bonded to a package input/output pin and are not available for in-circuit tests which must be conducted by some other method.

There are many well documented test approaches that can be applied to Gate Arrays, Standard Cells, and Fully Custom integrated circuits and the designer must evaluate those approaches or derive new ones.

Packaging

Digital systems implemented with the building blocks described in this paper are usually complex, because many functions are placed inside one device. This places a constraint on packaging since such complex devices require large numbers of input and output connections. It is not uncommon for a designer to discover that the chosen partitioning of the system places too many input and output connections on one device.

Manufacturers are working on various packaging approaches that provide large numbers of external connections. A designer who desires to place one to two hundred connections to a single device must consider the limitations of packages available for such applications. Additionally, any impacts on the producibility of the printed circuit card assembly must be evaluated before a commitment is made to such a design. Programs presently under way, such as the Department of Defense's Very High Speed Integrated Circuit program, have proven the feasibility of such devices, but each application requiring such packages must be evaluated individually.

Approaches to provide interconnections in these complex systems include the placement of multiple integrated circuits inside one package. This approach, the hybrid approach, has been taken a step further to include the placement of several different integrated circuits side-by-side on the same wafer and actually uses the lithography process to produce the digital systems on the wafer. Constraints such as yield, heat dissipation, and final system packaging make this a challenging approach, but some companies have already reported success.

Conclusion

Digital avionics designers have at their disposal powerful system building blocks and design tools. In order to efficiently use these capabilities, designers must evaluate the technology and design options in an informed manner early in the program. As shown by the large number of manufacturers in the VLSI business, many systems have benefitted from these technologies and design approaches. As the VLSI industry matures, Gate Array and Standard Cell sizes will be increased, better packaging methods will be developed, and system design tools will be improved and refined, thus making it feasible for VLSI technology to be used for many additional applications.

VLSI DESIGN FOR TESTABILITY

E.J.McCluskey*
The CENTER for RELIABLE COMPUTING
Computer Systems Laboratory
Depts. of E.E. and C.S.
Stanford University
Stanford, CA

Abstract

This paper presents a survey of "Design for Testability" (DFT) techniques. Emphasis is placed on those techniques currently in use for VLSI or those most promising for future adoption. The various scan path techniques are described and contrasted. Modifications of these techniques for use in BIST (built-in self test) applications are discussed. The most promising approaches for modifying array logic structures (PLAs, PALs, etc.) to make them testable are surveyed and evaluated.

Introduction

Testing of individual SSI, MSI, and LSI chips isn't a major problem. For systems implemented using such devices, board testing is the major difficulty. Two approaches have been relied on to control the cost of testing such boards: use of "bed-of-nails" fixtures and adherence to "design-for-testability" rules, [Williams 83], [Turino 79].

Semi-custom logic - gate arrays, standard cells, and logic arrays - are used for product volumes greater than that suitable for catalog part implementation but not sufficiently large to justify a full custom design. The minimum-area possibility of a custom design is given up in order to reduce design time and cost. Test vector development is a critical component of the design time. The incorporation of general features in the design to enhance testability - design for testability - can reduce drastically the time required to develop tests and increase substantially the effectiveness of the resulting test procedure. These testing benefits are obtained at the cost of additional chip area. By careful choice of the design-for-testability (DFT) techniques, the area overhead can be held to a small value. This paper presents a survey of the DFT techniques available for semi-custom designs.

There are two classes of tests that must be carried out on a device: parametric tests that determine whether the device parameters such as delay, impedances, etc. are within specifications and functional tests that test for the presence of faults that prevent the device from realizing the correct function (nodes stuck at a constant value, leads stuck open, two nodes incorrectly connected together, etc.). Parametric tests are necessary and important, but are not a major design problem as are functional tests: the major emphasis here is on functional tests. Two types of functional tests are important: (1) fault detection tests, also called go/no go tests and (2) fault diagnosis tests, also called fault location tests. Fault diagnosis tests are used after a part has failed a fault detection test and it is desired to determine the faulty element. The emphasis here is on fault detection tests.

Testing cost is determined by the cost of generating and maintaining test patterns, and by the cost of test application. The major test pattern generation cost factors are: (1) the computer time required to run the test pattern generation program plus the (pro-rated) capital cost of developing the program or the number of man hours required for a person to write the test patterns, and (2) the computer time for fault simulation to determine the fault coverage of the test patterns. The fault coverage computation is called fault grading and is often included as part of the automatic test pattern generation program. Test application cost is determined by the cost of the test equipment and the tester time required to apply the test (sometimes called socket time).

The most troublesome aspect of functional testing is test pattern generation, the selection of the inputs to be applied during testing. Table 1 lists the different strategies that are used for test pattern generation.

Table 1 Test Pattern Generation Methods

1. Use of Design Verification Inputs
2. Manual generation by designer or test engineer.
3. Fault-free Simulation to verify that each node is "toggled": has both 0 and 1 applied to it.
4. Use of Boolean Difference.
5. Use of a path-tracing algorithm (D algorithm) to generate tests for single stuck faults.
6. Single-stuck-fault simulation to select from a set of randomly generated input patterns.
7. Exhaustive and Pseudo-exhaustive testing: all possible input patterns are applied to the entire network or to network segments.
8. Pseudo-random testing: a long sequence of pseudo-random input patterns is applied.

Method 1, use of the design verification input patterns as production test inputs, has sometimes been adopted for large-volume, full custom designs. With this technique the set of test patterns are augmented on the basis of user experience with the part. A good set of test patterns is not obtained until after many chips have been shipped. This technique is not recommended for semi-custom parts with their more limited production runs. Manual test pattern generation (Method 2) is closely related. It has the disadvantages of requiring a long time (perhaps many months) to obtain the tests and also requires a large number of man hours. Coverage is usually low.

Method 3, the node wiggle approach, usually leads to very low fault coverage. The density of modern semi-custom parts requires high values of fault coverage in order to obtain small values of defective shipped parts. The fault coverage required depends on the yield as well as the desired defective part objective.

*Fellow, IEEE

The Boolean Difference (Method 4) is an algebraic technique, useful for theoretical studies rather than for production test generation. Path tracing (Method 5) is a more practical implementation of the same general approach.

Methods 5 and 6 can produce high fault coverage, but are very expensive in computer time. Very substantial reductions in this expense can be obtained by the use of scan-path techniques (to be discussed subsequently) to permit only combinational circuit test pattern generation. Statistical sampling techniques, [Agrawal 81], can be used to reduce the cost of the fault simulation.

Methods 7 and 8 will be discussed in connection with built-in-self-test techniques.

Design for Testability

The number of devices that can be fabricated on a single VLSI chip has been increasing each year without a proportional increase in the number of I/O pins. This density growth aggravates the already difficult problem of testing such microcircuits and has lead to a widespread belief that economical testing can only be obtained if a testing strategy is adopted during the initial chip design. The name "design for testability" or DFT has been adopted for those design techniques used to enhance chip "testability." The available DFT methods are listed in Table 2.

Table 2 Design for Testability Methods

Ad hoc Techniques: Design rules listing structures that cause testing problems and techniques for avoiding the problems.
Testability Measures: Programs that analyze a design and estimate the difficulty of test pattern generation as a measure of testability.
Testable structures: Design structures that are easily tested.
Scan Techniques: Techniques to realize internal bistable elements for easy access.
Built-in Self Test: Chip architectures that incorporate self-test ability.

Ad Hoc Methods

The earliest design for testability, DFT, method was a collection of ad hoc guidelines to be followed in carrying out a design, [Writer 75]. This approach has the disadvantages of (1) adding to the concerns of the designer, and (2) depending on the skill of the designer. More modern DFT techniques are structured. They enforce a structure on the design by means of design rules. Presentations of the ad hoc approach can be found in [Williams 83] and [Turino 79].

Testability Measures

Attempts to understand circuit attributes which influence testability have produced the two concepts of observability (visibility) and controllability (control), [Chang 74]. Observability refers to the ease with which the state of internal signals can be determined. Controllability refers to the ease of producing a specific internal signal value. The testability measure programs are based on approximating these attributes with specific numeric values. In particular, they attempt to predict the difficulty of doing test pattern generation for a particular design. They also attempt to point out structures in the circuit that cause testing problems in the hope that the circuit can be redesigned to improve its testability. An evaluation of these techniques is presented in [Agrawal 82] along with a list of relevant references.

Scan Techniques

Most of the DFT techniques are attempts to increase the observability or controllability of a circuit design. The most direct way to do this is to introduce **test points**, that is, additional circuit inputs and outputs to be used during testing. While this is a useful technique at the board level, very few I/O pins can be devoted to testing at the IC level. The effect of many test points can be obtained by incorporating test circuitry internal to the chip that permits access to many internal nodes through a very few I/O pins. In order to do this it is necessary to enter test data and read out test results in a serial fashion, that is, the test data is "scanned" or shifted in and out of the IC. There are a number of DFT structures for providing internal test access that are based on this approach. DFT methods based on these structures are called scan techniques. The scan techniques are general in the sense that they are applicable to random logic circuitry - they make no assumptions about the structure of the circuit other than that it is composed of logic gates and bistable elements.

Most of the scan techniques convert all of the internal bistable elements to test points. Doing this provides another very important benefit: test patterns need only be generated for the combinational circuitry. The bistable elements are tested directly by means of the test pins.

Another method for providing test access involves multiplexing test data onto some of the normal I/O pins during testing. For most technologies this is more costly than using a scan method and is thus used much less often.

Scan Path Techniques

The most common forms of scan techniques are based on converting the system bistable elements into a shift register called a scan path. With the circuit in test mode it is possible to shift an arbitrary test pattern into the bistable elements. By returning the circuit to normal mode for one clock period, the combinational circuitry acts upon the bistable element contents and primary input signals and stores the results in the bistable elements. If the circuit is then placed into test mode, it is possible to shift out the contents of the bistable elements and compare these contents with the correct response.

It will be assumed initially that the circuit is constructed of D flip-flops interconnected by combinational circuits and that all of the flip-flops are controlled by a single system clock. These assumptions mean that the circuit can be considered to have the general structure shown in Fig. 1. Drawing the circuit in this form is done to simplify the following discussion, but is not meant to imply any restrictions on the designer other than those just stated. The same general results hold true if other types of flip-flops (S-R, J-K, T...) are used. Since the extension to other flip-flops is straightforward, the details will not be presented here. Techniques for systems using latches rather than flip-flops will be

described later.

In the following discussion of various scan path techniques it is important to distinguish between latches and flip-flops. A latch has the transparency property: if the data input changes while the clock is active, the latch output will follow the data input change. A flip-flop changes only when the clock input makes a specific transition - the active transition. The flip-flop output takes on the value present at the data input when this active transition occurs. Subsequent changes of the data input have no effect until the next active transition of the clock. An important characteristic of flip-flops is that a shift register can be constructed by connecting the output of one flip-flop directly to the data input of the next flip-flop; the conversion of a latch register to a shift register requires an extra latch between each register stage. The symbols used for bistable logic elements follow IEEE Standard 91-1984.

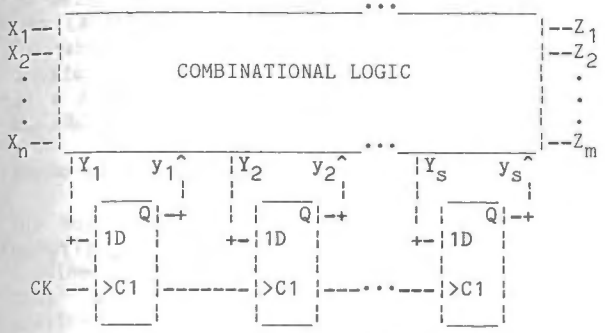


Figure 1 General structure of circuit for scan path discussion.

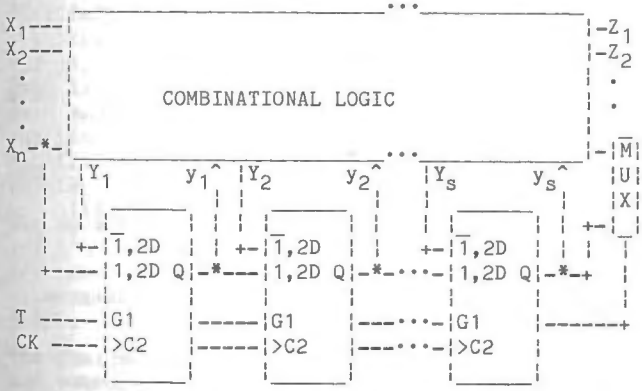


Figure 2 Stanford scan-path architecture - MD flip-flops used to provide scan path.

Stanford Scan Path Design

The first published description of a scan-path design-for-testability structure was [Williams 73], a paper based on the Stanford Ph.D. research of Michael Williams. In this technique each of the circuit flip-flops is replaced by a Multiplexed Data or MD flip-flop as shown in Fig. 2. A multiplexer is placed at the data input to permit a selection of two different data inputs - $\overline{1}$,2D (normal system operation) and 1,2D (test mode). The choice of data input is based on the value of the control input, T. When T=0, data is gated from the $\overline{1}$,2D input upon an active clock transition. Data is taken from 1,2D if T is equal to 1.

The modification of the basic circuit structure of Fig. 1 to obtain a scan path architecture using MD flip-flops is shown in Fig. 2. One additional input, the T input, has been added. For normal operation, T is equal to 0 and the circuit is connected as in Fig. 1. The upper data inputs ($Y_1 \ldots Y_s$) act as the flip-flop D inputs. In order to test the circuit, T is set equal to 1. The lower data inputs become the flip-flop D inputs. Thus $D_i = Q_{i-1}$ for i from 2 to s, and a shift register is formed. The primary input X_n is connected to D_1 becoming the shift register input and Q_s, the shift register output, appears at the primary output Z_m.

Testing of the combinational logic is accomplished by: (1) Setting T = 1 (Scan Mode); (2) Shifting the test pattern y_j values into the flip-flops; (3) Setting the corresponding test values on the X_i inputs; (4) Setting T = 0 and, after a sufficient time for the combinational logic to settle, checking the output Z_k values; (5) Applying a clock signal to CK; (6) Setting T = 1 and shifting out the flip-flop contents via Z_m. The next y_j test pattern can be shifted in at the same time. The y_j values shifted out are compared with the good response values for y_j. The flip-flops must also be tested. This is accomplished by shifting a string of 1's and then a string of 0's through the shift register to verify the possibility of shifting both a 1 and a 0 into each flip-flop.

Two-port Flip-flop Designs

A basic requirement of the scan path technique is that it be possible to gate data into the system flip-flops from two different sources. One method of doing this is to add multiplexers to the system flip-flops as shown in Fig. 2. Another possibility is to replace each system flip-flop by a two-port flip-flop, a flip-flop having two control inputs with the data source determined by which of the controls is pulsed. When a pulse is applied to C1, data is entered from D1; and when a pulse occurs at C2, data is entered from D2.

Figure 3 shows the structure of a network with two-port flip-flops used to provide the scan path. The testing procedure is basically the same as that described in connection with Fig. 2. In the circuit of Fig. 3, changing between test mode and normal mode is accomplished by changing the clocking rather than by changing the mode signal.

Latch-based Structures

Some systems are designed using latches rather than flip-flops as the bistable elements. For latch-based systems it is not possible to directly reconfigure the system bistable elements into a shift register for test purposes. The most popular technique for introducing scan path testability into latch-based systems is IBM's LSSD (Level Sensitive Scan Design). This requires the use of extra latches to allow the system latches to be connected into a shift register. The Univac Scan-set technique, [Stewart 77] and [Stewart 78], avoids the necessity of configuring the system latches into a a shift register by using a separate test data shift register. This register can load test data in parallel to or from the system latches. Univac has also proposed the use of multiplexers to scan out latch contents. Fujitsu

and Amdahl avoid the use of test shift registers entirely. They use a combination of demultiplexing and multiplexing to set and scan out the system latches.

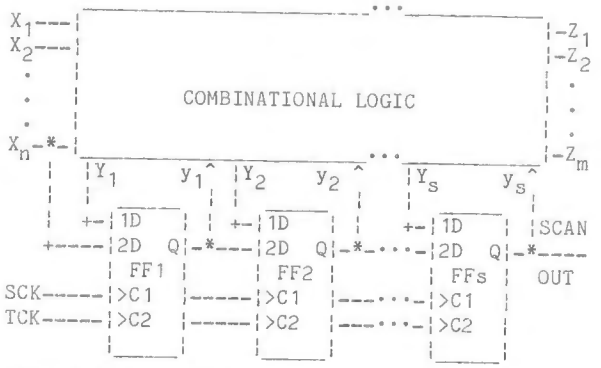


Figure 3 General structure of circuit using two-port flip-flops to provide scan path.

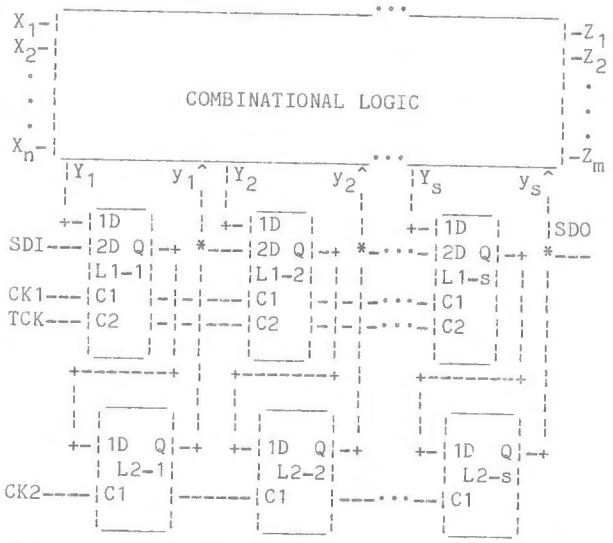


Figure 4 General structure of circuit using two-port latches to provide scan path - LSSD double latch design. SDI is scanned-in test data, SDO scanned-out test data. TCK is test clock.

Level-sensitive Scan Design - LSSD

A scan path design method, called level-sensitive scan design (LSSD), for latch-based systems was presented in [Eichelberger 77]. It is the standard design technique in current use at IBM. In this method each system latch is replaced by a two-port latch (L1 latch), and a second (single-port) latch (L2 latch) is added to permit reconfiguration of the system latches into a shift register for test purposes. A two-port latch is directly analogous to a two-port flip-flop. It is a latch with two data inputs each of which is controlled by a separate clock. The reason that this method is called level-sensitive has to do with the design of the latches to be hazard free and thus not dependent on the clock rise and fall times for correct operation.

A general structure for a system designed using the LSSD technique is shown in Fig. 4. Examples of some specific designs using this structure are presented in [Das Gupta 78]. During normal operation the system is clocked with two interleaved non-overlapping pulse trains applied to the CK1 and CK2 inputs. Other possible structures are discussed in [Eichelberger 77]. The way a system designed using this technique is tested is very similar to testing a system using two-port flip-flops. The differences when using LSSD are: The test vector y_j values are scanned in via SDI by applying pulses alternately to the test clock input TCK (called A clock in some LSSD papers) and the system clock input CK2 (also called B clock). The new values of y_j are entered into the corresponding L1 latches by applying one clock pulse to the system clock CK1. The new y_j values are scanned out by applying clock pulses alternately to CK_2 and TCK.

Random Access Scan Design

Fujitsu and Amdahl use the principles of multiplexing and demultiplexing to implement a scan technique for latch-based systems, [Ando 80], [Wagner 83]. A simplified version of the latch design used is shown in Fig. 5. This is called an addressable latch. Inputs 1D(Yi) and C1(CK) are used during normal system operation. In order to access Latch i for test purposes, the signal "Select i" must be set to 1. With Select i = 1, the latch content is placed on SDO(i), and SDI is clocked into the latch if TCK is pulsed. The structure of a system using these latches is shown in Fig. 6.

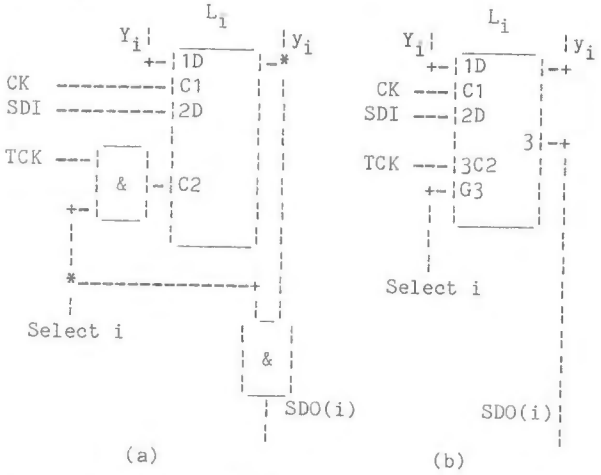


Figure 5 Addressable latch. (a) Circuit diagram. (b) Symbol.

There is associated with the circuit an address register whose contents are decoded to produce the "Select i" signals. At most one of these signals is equal to 1 at a time. Data is scanned into the latches by placing the latch i data value on SDI, the i address in the address register, and then pulsing TCK. The address register is implemented as a counter. Thus a sequence of data can be scanned into the latches by placing the sequence on SDI and pulsing the address register counter and TCK in the proper time relationship. The latch contents are scanned out via SDO by pulsing the address register to select the latches in turn.

An important feature of this structure is the ability to scan out the latches during normal system operation. The actual implementations of this technique using addressable latches have two or three select signals per latch. These signals are decoded at each latch using the AND gate. Somewhat more complex latches are used in the

actual systems in order to take advantage of the technology (ECL) and minimize the penalties due to addressability. These are described in [Ando 80] and [Wagner 83].

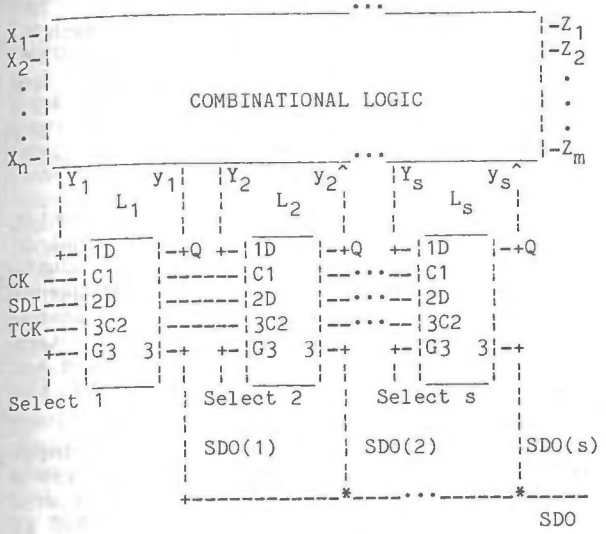


Figure 6 General structure of circuit using random access scan.

The External Scan Path

The scan path techniques just described make the test pattern generation problem much easier: combinational circuit tests are simpler to derive than sequential circuit tests. The test application difficulties still remain. The amount of test data that must be stored is reduced, but the time to apply a single pattern is increased due to the need to scan data in and out of the chip. A considerable reduction in tester complexity can be obtained by adding an external scan path to the design, [Zasio 83]. The external or boundary scan path is formed by associating a flip-flop or latch pair with each of the I/O bonding pads. The resulting structure is shown in Fig. 7 and the details of the circuitry associated with the bonding pads are shown in Fig. 8.

The advantages of the external scan ring structure are: 1. It is not necessary to probe all of the I/O pins during wafer sort. 2. The board interconnections can be easily tested. 3. A simple test of the operating speed is possible.

The Zasio paper describes the testing of a 256 pin chip by probing only 13 pins during wafer sort. Seven of the probed pins are used only for test: two test clocks, three mode controls, a scan-in and a scan-out pin. (Two test clocks are required since the design is latch-based and uses a two-phase clock.) The other probed pins are power, ground, and up to four system clocks.

The testing of the functional circuitry is done by shifting the test patterns into the internal and external scan paths and using the system clock to transfer the new state into the internal path. The new state and output response are transferred off the chip by means of the external and internal scan paths.

The board interconnections can be tested by shifting data onto the output pads via the external scan path, capturing the signals present at the input pads in the corresponding external scan path flip-flops, and then shifting out the captured data to verify that the correct signals were received.

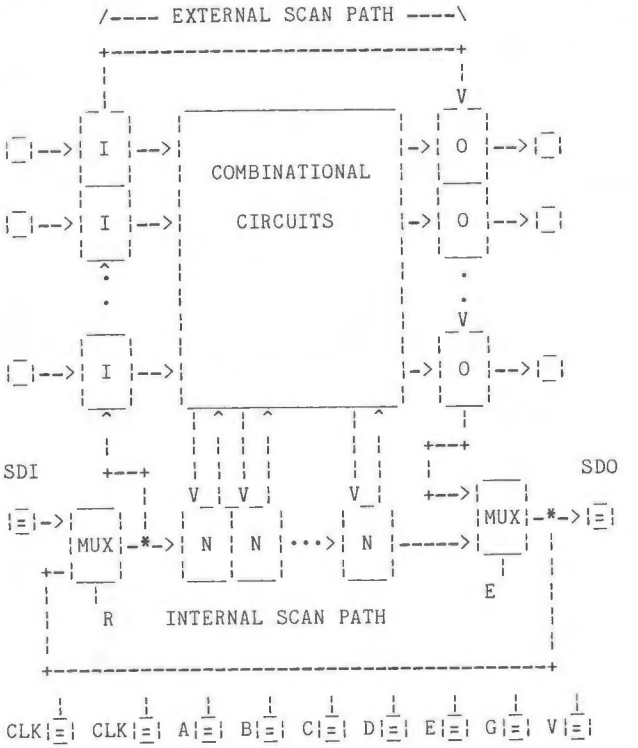


Figure 7 Structure with both internal and external scan paths.

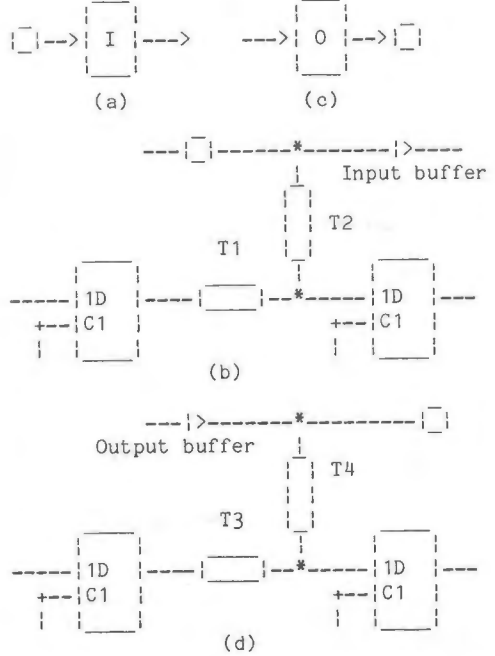


Figure 8 Details of I/O bonding pad scan path elements. (a) Input pad symbol. (b) Input pad circuit. (c) Output pad symbol. (d) Output pad circuit.

The chip speed is checked by connecting the internal path output back to its input by means of the two multiplexers shown in Fig. 7. An inversion is provided so that a ring oscillator is formed. By measuring the speed of oscillation, a measure of the chip operating speed is obtained. A

similar technique is also used on the external scan path. Other discussions of the use of an I/O scan path are presented in [Komonytsky 82] and [Resnick 83].

Other Scan Path Techniques

A number of extensions to the basic scan-path technique have been proposed to extend its usefulness.

Imbedded Memory

It is becoming increasingly common to have memory (ROM or RAM) included as part of a circuit. Testing of such designs using scan-path techniques is not straightforward. A modification of the basic LSSD design method to permit efficient testing of circuits with imbedded memory is presented in [Eichelberger 78]. Different methods for testing such circuits are discussed in [Funatsu 78] and [Ando 80]. In these papers the embedded memories are called "arrays", although the techniques described are not intended to apply either to programmed logic arrays or to gate arrays. A scheme for a built-in-self test of embedded memory is described in [Sun 84].

Miscellaneous Extensions

The use of a random test pattern generator to obtain tests for an LSSD design is described in [Williams 77]. Modifications to the basic LSSD latch design for a variety of applications including those in which an LSSD design is interfaced with non-LSSD circuitry are presented in [Das Gupta 81].

Testable Structures

Some types of structure occur very frequently and thus warrant special attention. Two such types of structure are the PLA and the counter or frequency divider. Techniques have been developed for implementing these structures in an easily testable fashion.

The PLA is such a common structure in current VLSI designs that a number of techniques have been proposed for modifying the basic PLA structure so that it is easy to test. The common feature of all of the easily-testible PLA schemes is the ability, during test, to select individual word (product) lines and to verify that the correct crosspoints are present. The easily-testable PLA schemes fall into two broad categories: (1) the universal or function-independent testing schemes that have test sets that depend only on the size of the PLA and not on the particular functions implemented, and (2) the function-dependent schemes that have test sets that are specific to the particular PLA design.

A common feature of the universally testable designs is the use of a shift register to select the product lines one at a time. One such scheme in which a comparator is used to verify the correct response is described in [Fujiwara 84]. Since the test patterns are function-independent and regular, they could be generated by on-chip circuitry to obtain built-in self-testing.

If function-independent testing is not a requirement it is possible to avoid the area overhead of the shift register. The scheme presented in [Bozorgui-Nesbat 84] makes use of a small number of additional bit lines that are used only during testing. The added bit lines permit isolation of each word line and verification of the crosspoints. A program for identifying the required additional test bit lines and the corresponding test patterns is described. This technique requires less area overhead and has higher fault coverage than the other methods proposed for easily-testable PLAs. It has the additional advantage of not requiring any alteration in the basic PLA structure and is thus useful for testing catalog part PLAs or PALs. Its drawback is its unsuitability for built-in self-test.

Some semi-custom chips contain extensive counter circuitry. Most of these counters are not required to deliver a true binary number for the count. Many counter structures are used for frequency division and need only provide an indication that a certain state has been reached. Other counters contain memory addresses. These addresses need not correspond to the correct binary count of the number of pulses received.

When a correct binary count of the number of input pulses is not necessary, it is desirable to use a Linear Feedback Shift Register, LFSR, rather than a true binary counter. The advantage of the LFSR is that it can be reconfigured into a true shift register for scan path testing without replacing each flip-flop with a dual port or multiplexed input flip-flop. The test reconfiguration can be accomplished by placing a multiplexer only at the first stage flip-flop. A description of the use of counters in connection with design-for-testability is presented in the following paper of this session: "Design for Testability in VLSI Chips: A Case Study."

Built-in Self Test Architectures

Several schemes for incorporating BIST techniques into a design have been proposed. They differ in the requirements placed on the functional circuitry and on the mechanisms used to generate test inputs and analyze test output response. The different possibilities are listed in Table 3. Not all combinations of mechanisms are viable; for example, if the functional circuit doesn't have scan paths, it isn't possible to reconfigure the scan path into a test pattern generator. A discussion of the details of several different BIST architectures follows.

Table 3. Choices for BIST Architectures

Functional Circuitry	
	No Scan path
	Internal scan path only
	Both Internal and External Scan paths
Input Test Pattern Generation	
	Done in separate chip on the same module
	Internal Scan Path Reconfigured
	External Scan Path Reconfigured
Output Response Analysis	
	Done in separate chip on the same module
	Internal Scan Path Reconfigured
	External Scan Path Reconfigured
	Done by Checking Circuits

The only commercial semi-custom part with BIST currently announced uses an external scan path that is reconfigured for test stimulus and response analysis. No scan path is required in the functional circuitry. This part is the SCX6260

CMOS Gate Array of National Semiconductor. The design of this array was done at CDC, [Resnick 83], and is produced also by Motorola.

External test chip and no scan paths.

The earliest detailed study of a BIST structure for random logic chips is described in [Benowitz 75]. Each module has a response analysis circuit and an input multiplexer added to it. The multiplexer chooses between the normal system inputs and an external LFSR that generates test patterns. A number of possibilities were studied for the response analyzers. The results of simulation studies that compare cost, test time, and fault coverage for several implementations are presented in the paper. Parity check circuits were found to be the fastest but also the most expensive and poorest in fault coverage. This paper was the first to suggest the use of an LFSR for output response analysis. Both serial and parallel implementations were simulated. In the simulations, the fault coverage was determined by simulating stuck faults at circuit I/O's. The only assumption made about the circuit-under-test is that it does not contain RAM. Sequential as well as combinational logic circuits can be present in the circuit-under-test.

A similar structure is described in [Perkins 80]. This structure makes use of a tester circuit that is external to the chip, but could be located on the same board. Pseudorandom patterns are applied in parallel to the chip inputs and a parallel signature analyzer receives both the chip outputs as well as the test inputs. A special feature of this design is its ability to be configured by means of a tester circuit register to apply input signals to the appropriate chip input pins. Testing of RAM and ROM chips is discussed. The most novel feature of the structure of [Perkins 80] is the inclusion of a source resistance detector at each logic input and an output current detector at each logic output. The main purpose of these detectors is to watch over the integrity of the interconnection system.

Internal scan path and external test chip.

The inclusion of an internal scan path in the functional circuitry permits more careful control of the fault coverage. A design for BIST structure for chips with internal scan paths mounted on a multi-chip module is presented in [Bardell 82]. This design uses a special test chip added to the module. The test chip contains a shift register pattern generator and a parallel signature analyzer. The scan paths on each chip are loaded in parallel from the pattern generator. The system clocks are then pulsed and the test results are scanned out to the parallel signature analyzer. New test patterns can be scanned in at the same time that the test results are being scanned out.

Internal and External scan paths with an external test chip

A drawback of the previous scheme is the difficulty of testing the interconnections, either those between the chip pins and bonding pads or those between chips. This is made much easier by the inclusion of an external or boundary scan path on the chip. An external scan path consists of a scan path flip-flop connected to each of the bonding pads. A structure that makes use of an external scan path in connection with an external test chip was proposed in [Eichelberger 83] and is shown in Figure 9. The scan input of the external scan path is connected to the scan out point of the internal scan path. Pseudo-random test patterns are generated in a pattern generator circuit and are scanned into the combined scan path. The system clocks are pulsed and the scan path latches are scanned out into the serial signature analyzer circuit. The resulting signature is then compared in the analyzer with a precalculated fault-free circuit signature in order to generate a failure signal. The scan-out point is also connected to a pin so that, in case of a failure, intermediate signatures can be examined externally for diagnostic purposes.

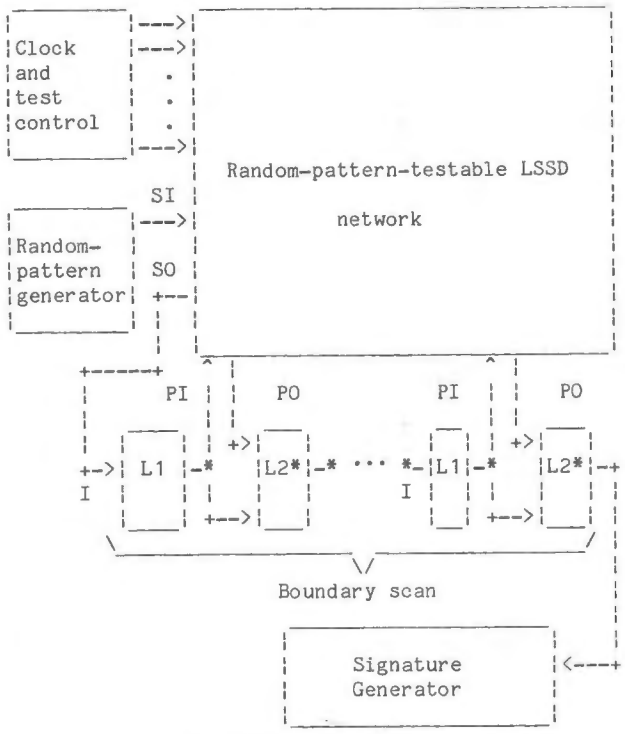


Figure 9 BIST structure using internal and external scan paths in conjuction with an external test chip.

BIST Structure using register reconfiguration.

A concern with BIST designs is the amount of extra circuitry required. One technique for reducing the extra circuitry is to make use of the flip-flops or latches already in the design for test generation and analysis. The system registers are redesigned so that they can function as pattern generators or signature analyzers for test purposes.

The structure described in [Konemann 79] and [Konemann 80] applies to circuits that can be partitioned into independent modules. Each module is assumed to have its own input and output registers, or such registers are added to the circuit where necessary. No precise definition of a module is given, nor is the problem of identifying modules discussed. It is assumed that the circuit modularity is evident. The registers are redesigned so that for testing purposes they can act as either shift registers or parallel signature analyzers. The redesigned register is called a BILBO (Built-in Logic Block Observer).

The technique just described is most suitable for circuits that can be partitioned so that the input and output registers of the resulting modules can be reconfigured separately. They can be used for scan path designs, but the existence of scan paths is not a requirement. There are other reconfiguration techniques that are suitable for scan path designs.

BIST Structure using external scan-path reconfiguration.

A technique in which the external scan path is reconfigured into an LFSR for test pattern generation and into another LFSR for signature analysis is described in [Komonytsky 82], [Komonytsky 83] and [Resnick 83]. The basic design of these circuits incorporates both an internal scan path and an external scan path for the I/O pads. The BIST modification provides for the reconfiguration of the input scan path latches into an LSFR for use as a pseudorandom pattern generator and the reconfiguration of the output scan path latches into a signature analyzer. A test pattern is generated in the LFSR, scanned into the internal latches, the system clock is pulsed, and the resulting latch contents are scanned out to the signature analyzer. The details of the design are presented in the references cited.

Other BIST designs using LFSR's for test vector generation and signature analysis are described in [Eiki 80], [Heckelman 81], and [Fasang 80].

BIST Structure using concurrent checking circuits.

For systems that include concurrent checking circuits, it is possible to use this circuitry to verify the response during explicit (off-line) testing. Thus the necessity of implementing a separate response analysis circuit such as a signature analyzer is avoided. This approach to BIST is described in [Sedmak 79], [Sedmak 80].

Conclusions

Design for testability features are vital for semi-custom parts due to the large number of different designs that must be produced. The cell library should contain optimized designs for multi-port and multiplexer input flip-flops to permit the establishment of a scan path or paths without large area overhead. Also included in the cell library should be counter structures based on the LFSR with the appropriate scan-path features included. The PLA design facility should include the ability to produce automatically an easily testable structure with the appropriate test vectors. In the future other DFT features may be identified, perhaps built-in self test structures. For the present the facilities just described are the minimum required for adequate economic testing of semi-custom chips.

Acknowledgements

The preparation and presentation of this paper was supported in part by the National Science Foundation under Grant No. MCS-8200129 and in part by the Lockheed Missiles and Space Company, Inc. The author would like to thank Joseph McCluskey for his help in preparing the manuscript.

References

[Agrawal 81] Agrawal, V.D., **Jrnl. of Digital Systems**, Vol.V, No. 3, Fall 1981, pp. 189-202.

[Agrawal 82] Agrawal, V.D., **IEEE Test Conf.**, pp. 391-396, 1982.

[Ando 80] Ando, H., **COMPCON Spring 80**, San Francisco, CA, pp. 50-52, Feb. 25-28, 1980.

[Bardell 82] Bardell, P.H., **IEEE Test Conf.**, Philadelphia, PA, pp. 200-204, Nov. 11-13, 1982.

[Benowitz 75] Benowitz, N., **IEEE Trans. Comput.**, c-24, No. 5, pp. 489-497, May 1975.

[Bozorgui-Nesbat 84] Bozorgui-Nesbat, S., **IEEE Test Conf.**, 1984.

[Chang 74] Chang, H.Y., **BSTJ**, pp. 1505-1534, Oct. 1974.

[Das Gupta 78] Das Gupta, S., **1978 IEEE International Solid-State Circuits Conf.**, San Francisco, CA, pp. 216-217, Feb. 15-17, 1978.

[Das Gupta 81] Das Gupta, S., **Eleventh Ann. Int'l Symp. on Fault-Tolerant Computing, (FTCS-11)**, Portland Maine, pp. 32-34, June 24-26, 1981.

[Eiki 80] Eiki, H., **10th Int'l Symp. on Fault-Tolerant Computing, (FTCS-10)**, Kyoto, Japan, pp. 173-178, Oct. 1-3, 1980.

[Eichelberger 77] Eichelberger, E.B. **14th Design Automation Conf.**, New Orleans, LA, pp. 462-468, June 20-22, 1977.

[Eichelberger 78] Eichelberger, E.B., **3rd USA-JAPAN Computer Conf.**, San Francisco, CA, pp. 266-272, Oct. 10-12, 1978.

[Eichelberger 83] Eichelberger, E.B., **IBM J. Res. Develop.**, Vol. 27, No. 3, pp. 265-272, May 1983.

[Fasang 80] Fasang, P.P., **IEEE Test Conf.**, Philadelphia, PA, pp. 261-266, Nov. 11-13, 1980.

[Fujiwara 84] Fujiwara, H., **IEEE Trans. Comput.**, c-33, No.8, pp. 745-750, Aug. 1984.

[Funatsu 78] Funsatsu, S., **IEEE Test Conf.**, Cherry Hill, NJ, pp. 98-102, Oct.31-Nov.2, 1978.

[Heckelman 81] Heckelman, R.W., **IEEE Solid State Circuits Conf.**, pp. 174-175, Feb. 19, 1981.

[Komonytsky 82] Komonytsky, D., **IEEE Test Conf.**, Philadelphia, PA, pp. 414-424, Nov. 15-18, 1982.

[Komonytsky 83] Komonytsky, D., **Electronics**, pp. 110-115, Mar. 10, 1983.

[Konemann 79] Konemann, B., **IEEE Test Conf.**, Cherry Hill, NJ, pp. 37-41, Oct. 23-25, 1979.

[Konemann 80] Konemann, B., **IEEE J. of Solid-State Circuits**, Vol. SC-15, No. 3, pp. 315-318, June 1980.

[Perkins 80] Perkins, C.C., **IEEE Test Conf.**, Philadelphia, PA, pp. 29-41, Nov. 11-13, 1980.

[Resnick 83] Resnick, D.R., **VLSI Design**, pp. 34-38, March/April 1983.

[Sedmak 79] Sedmak, R.M., **IEEE Test Conf.**, Cherry Hill, NJ, pp. 112-124, Oct. 23-25, 1979.

[Sedmak 80] Sedmak, R.M., **IEEE Test Conf.**, Philadelphia, PA, pp. 267-278, Nov. 11-13, 1980.

[Stewart 77] Stewart, J.H., **IEEE Test Symp.**, Cherry Hill, NJ, pp. 6-15, Oct. 25-27, 1977.

[Stewart 78] Stewart, J.H., **IEEE Test Conf.**, Cherry Hill, NJ, pp. 152-158, Oct. 31-Nov. 2, 1978.

[Sun 84] Sun, Z., **IEEE Test Conf.**, 1984.

[Turino 79] Turino, J., **Electronics Test**, pp. 18-20, April 1979.

[Wagner 83] Wagner, K.D., **COMPCON Spring 83**, San Francisco, CA, pp. 384-388, Feb. 28-Mar. 3, 1983.

[Williams 73] Williams, M.J.Y., **IEEE Trans. Comput.**, c-22, No. 1, pp. 46-60, Jan. 1973.

[Williams 77] Williams, T.W., **IEEE Test Conf.**, Cherry Hill, NJ, pp. 19-27, Oct. 25-27, 1977.

[Williams 73] Williams, M.J.Y., **Proc. IEEE**, pp. 98-112, Jan. 1983.

[Writer 75] Writer, P.L., **'75 ASCC Conf. Record**, pp. 84-7, Oct. 1975.

[Zasio 83] Zasio, J.J., **COMPCON Spring 83**, San Francisco, CA, Feb. 28-Mar. 3, 1983.

SESSION 19 DATA LINK SYSTEMS APPLICATIONS

Chairman:

Darlow G. Botha
AFWAL/AAAI

Dennis G. Evans
PME/PMA

This session considers the applications of extra-vehicle data link and external reference systems and their implications to the rest of the avionics suite and the operational capabilities of the aircraft.

THE NAVY'S TACTICAL DATA NETWORK

Walter J. Schoppe*

Naval Air Development Center
Communication Navigation Technology Directorate
Warminster, Pa. 18974

ABSTRACT

Since the early 1960's, the Navy has utilized digital data links in support of their tactical Communication, Command and Control systems. These systems continue to be widely used to support task force operations and although new systems are in the development phase it is unlikely that any major changes will occur before the mid-1990's. The purpose of this paper will be to describe those systems, the way they work, the platforms that are equipped and the method of interconnection. The systems to be discussed are the Naval Tactical Data System (NTDS), the Airborne Tactical Data System (ATDS) and the Marine Tactical Data System (MTDS). These three systems depend on links 4, 11 and 14 to transfer data among various Navy platforms. The data that is exchanged by these systems includes messages and bit fields from TADIL(Tactical Data Interchange Link) A and TADIL C. Included in the paper will be descriptions of the Networking techniques used by the participating platforms, the waveform modulation and the coding techniques. Links 4 and 11 support an automatic communications network which interconnects tactical data processing and sensor equipment to provide the participating platforms with tracking and position information of all friendly and non-friendly surface, sub-surface and airborne platforms. This information can then be used by the tactical commanders to allocate resources to combat enemy activities. The paper will conclude with a brief description of the typical system configurations on the various platforms.

INTRODUCTION

Todays Navy uses a network of four independent tactical data systems to exchange tactical information. The four systems are:

Link 4A/TADIL C
Link 11/TADIL A
Link 14
TADIL B

A fifth system, Link 16/TADIL J, is being developed and will partially replace the first two systems listed above. These systems are implemented on their respective platforms by a mix of computers, displays and communications equipment. Depending on the nature of the platform the equipment is identified as:

NTDS -- Naval Tactical Data System
ATDS -- Airborne Tactical Data System
MTDS -- Marine Tactical Data System

Figure 1 is an artists sketch of the overall Naval tactical data network showing typical interconnections. The system is essentially a group of digital computer networks that interconnect the major force elements of a Battle Group. These force elements include many aircraft, ships and land-based units. Each of which may consist of various platforms having unique capabilities, performing specific functions that depend on the various links and TADILS for coordinated Operations.

Before starting a detailed description of the Navy's Tactical Data System it is good to define some of the systems involved.

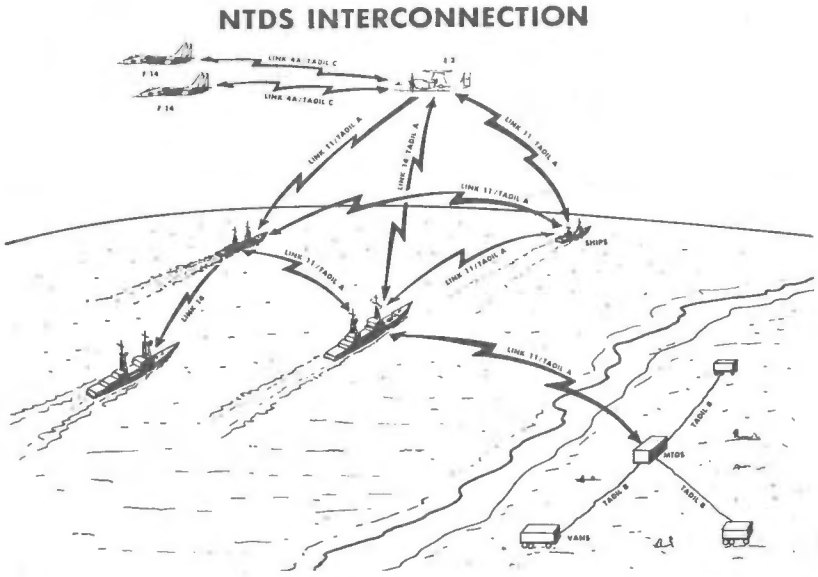


FIGURE 1

* Member IEEE

Link 4A/TADIL C-- Link 4A is the NATO designation of TADIL C. It is a one or 2-way nonsecure UHF data link between Air Control stations and controlled aircraft. It has its own unique set of message standards (frequently referred to as Control and Reply), and protocols, and does not directly interface with Link 11.

Link 11/TADIL A--Link 11 is the NATO designation of TADIL A. Link 11 is a tactical data information link employing netted communications techniques and standard message formats for the exchange of digital information among land based, submarine and ship tactical data systems. Link 11 is a one-half duplex netted normally secure digital data link that normally operates in a Roll Call mode under control of a Net Control Station. Link 11 operationally is automatic once initiated and provides for the mutual exchange of information among net participants via HF or UHF radios. Tactical data processing equipments located at each station are linked together via a common transmission medium into a netted configuration.

Link 14--Link 14 is the NATO designation of a one-way teletype link that provides computer controlled broadcast by Link 11 equipped units to other units.

TADIL B--TADIL B is a full duplex two-way point-to-point digital data link that provides for the serial transfer of data between units of the Army,Air Force, Marine Corps. and Special Information Systems. It has the same message standards as Link 11. The equipment, message protocols and data rates are different than Link 11, therefore messages cannot be directly interfaced with Link 11. Tadil B has no corresponding link designator.

Link 16/TADIL J--Link 16 is the NATO designation for the developmental link that is known as TADIL J. TADIL J combines and expands TADIL A and C message capabilities into one common message specification with standardized data elements. TADIL J may be used by existing Link 11 hardware or by JTIDS terminals which are under development. TADIL J is being designed so an indirect interface (data forwarding) can be maintained with a Model 4 Link 11 system for information exchange.

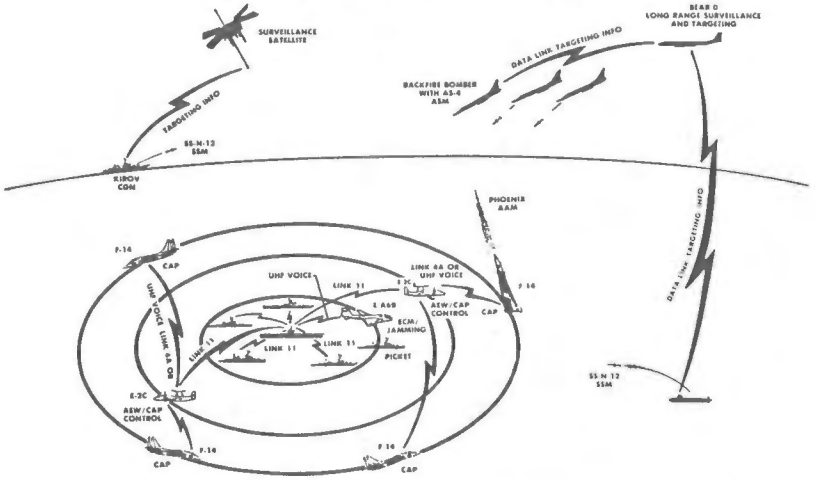
These systems supply the message and the communication links enabling the various elements of the Navy to exchange tactical data in a timely manner. Although the Navy performs many different missions these links are best identified with AAW and ASW where their implementation has been greatest. The value of the system derives from its ability to convey target data from one platform to another very rapidly. One of the primary features of the system is that the platforms are all equipped with their own unique computers and communication systems. Interoperability is made possible by standarization of the messages, the RF modulation and the data link protocols. The capability resulting from this systems provides the task force with the ability to coordinate activities such as target tracking in real-time, which results in a synergistic improvement in its operational effectiveness. Each Navy platform consists of a complex of sensors, computers, computer programs, personnel and communications equipments. The elements are integrated and organized to support Warfare Commanders in discharging their command responsibilities at sea. The objective of this paper is to describe the total system and how it is used by the Navy.

The two primary missions supported by the Tactical Data System of the Battle Group are Anti-Air Warfare and Anti-Submarine Warfare. A typical Battle Group might include a carrier, several destroyers, and several other support ships as well as the Carrier's complement of aircraft. The particular aircraft involved would depend on both the threat and the mission. Aircraft that most likely would be included are: the E-2C "Hawkeye" an Advanced Early Warning (AEW) platform, F-14A "TOMCAT", EA-6B "Intruder", an Electronic Warfare (EW) aircraft and the S-3A "Viking", an Anti-Submarine Warfare (ASW) platform. The aircraft with their mobility and speed are able to extend the coverage of the Battle Group to several hundred miles. An airborne enemy threat is responded to by launching E-2C's and F-14's while a submarine search requires S-3A's and ASW heocopters. Multiple threats may result in a mix of ASW and AAW platforms. These airborne platforms, as well as, whatever ships are included, are linked together by several communication systems which allow the platforms to update each other on position as well as to provide target and tracking inputs.

Figure 2 is a graphic representation of the Battle Group operating in an AAW posture. Surface ships and the E-2C's are linked together on Link 11 exchanging position and track data. Several F-14 interceptors are being controlled by the E-2C's via Link 4A. And finally non-NTDS ships are kept informed of the NTDS situation by Link 14. The incoming threat would be detected by either the E-2C or one of the picket ships. A report of the track would be broadcast to the NTDS network via Link 11 which would probably result in the assignment of an intercept to an appropriate AEW and his Combat Air Patrol. The AEW Commander would then assign one or more of his F-14A's interceptors to the threat and would guide him to an intercept point using the Link 4A data link. Between the E-2C and F-14, Link 4a has a two-way capability allowing the E-2C to control or vector the F-14 to some point and allowing the F-14 to update the E-2C with status and target information.

The third figure portrays the various participants in the Anti-Submarine Warfare scenario. The threat is now an enemy submarine. The defense involves ASW ships, helocopters, submarines in direct support to the battle group and finally the S-3A aircraft. The flight profile of the S-3A permits their use at considerable distance from the Battle group center while helocoptors are used more in the immediate vicinity. The S-3A is connected to the surface platforms over Link 11 which is generally operated at HF frequencies (2 to 30Mhz), since the S-3A will frequently drop below the radio-horizon. Of

FIGURE 2 **BATTLE GROUP AAW TACTICAL COMMUNICATION OVERVIEW**



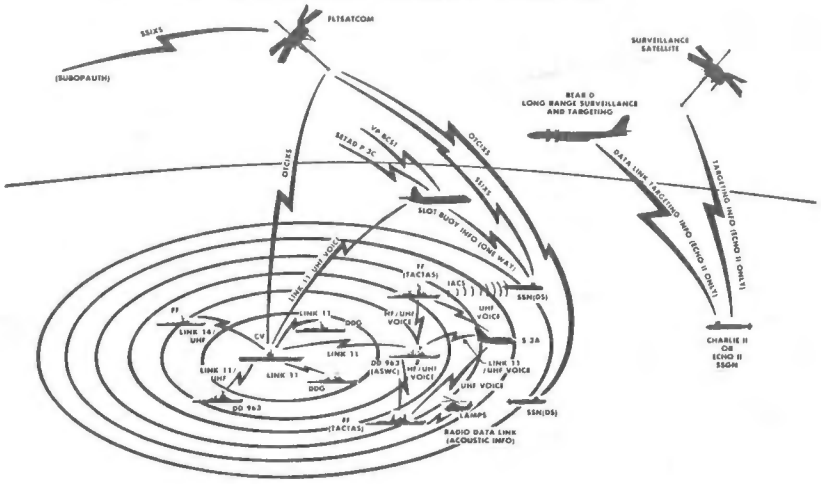
course any contacts made using the various ASW sensors are entered into the data system via Link 11. Frequently, involved in ASW operations are land-based P-3C's which perform the same ASW function as the S-3A's except their longer range and endurance allows them to be used at much greater distances from the task force where they are used to sweep the ocean well in front of the task force's operation area. One will notice several special data links which are not actually a part of the NTDS system in this figure one is an acoustical link between ships and submarines known as IACS (Integrated Acoustic Communication System) and the other is a high speed data link from the LAMPS Helocopter to its ship. The LAMPS platform is actually a tethered platform in that this data link is raw sensor data which is returned to the ship for processing and although the helocopter can communicate with other LAMPS capable ships it is strictly a point-to-point link. It should be noted that all of these data links are augmented with numerous voice links which permit clarification and interaction beyond the level available from the data links.

Link 4A/TADIL C

The Link 4A/TADIL C system normally interconnects tactical and support aircraft to the aircraft control units. TADIL C is a 5 kilobit-per-second half duplex digital data transmission system used to transfer aircraft control and target information between a control station and a controlled aircraft. The transmission is a serial Time Division Multiplex(TDM) signal on a single Ultra High Frequency (UHF) channel. The Navy's UHF band consists of 7000 channels ranging from 225.000 to 399.975 Mhz spaced by 25 Khz increments.

Link 4A was introduced operationally as an automatic landing system. The basic elements of the All-Weather Carrier Landing System (ACLS) consist of the data link, the aircrafts' autopilot and several aircraft displays while the system aboard the ship involves a radar and a computer. As the aircraft approaches the aircraft carrier, it is detected by radar (usually AN/SPN-42), which supplies the computer with precise position information. This information is then used to derive the ideal glide path for a successful carrier-deck landing. The computer then sends a formatted command message to a UHF transmitter. The command message, which is fixed in length, is sent as a frequency shift keyed UHF signal. The message contains a synchronization code, and data. For example, in the ACLS mode, the information might include pitch and yaw commands to the airplane. The transmitted message is received by a UHF receiver, decoded by the appropriate digital logic, and forwarded the appropriate

FIGURE 3 **ASW COMMUNICATIONS**



display or control device. The ACLS system has three modes of operation: Mode I (fully automatic); Mode II (semi-automatic-pilot controls to touchdown by observing cockpit indicators and display); and Mode III (talk down-shipboard controller provides verbal information for pilot to touchdown). In actual application a voice link usually accompanies the data link, although it is not necessary in Modes I and II. The displays normally found in the aircraft cockpit are a cross-pointer indicator and a descrete display panel where messages such as "ten seconds to touchdown" and "waveoff" are illuminated. While the aircraft is under positive control (Modes I and II) messages are sent at a rapid rate, continually updating the indicators and the autopilot. Imbedded in the message are several parity bits which provide an error detection capability; parity check should exist, with failure to satisfy the parity equation resulting in rejection of the entire message. At present, all carrier-based aircraft have the ACLS capability, usually in the form of the AN/ASW-25A receiver decoder.

Once the capability of providing accurate position information to aircraft was validated, many new applications were considered. such as vectoring aircraft from ground locations to airborne targets. As a result two new applications were developed: E-2C to F-14A data link and Carrier Aircraft Inertial Navigation System (CAINS) update. The E-2C "Hawkeye" to F-14A "Tomcat" data link provides the Navy with its most effective task force defense system. The E-2C is a five-man aircraft which is distinguished by a massive rotodome mounted above its fuselage. The rotodome is part of a RADAR system that allows the E-2C to detect incoming missiles or aircraft at distances of several hundred miles. Once a target is detected, the E-2C computer is able to automatically track and project its future location and send vectoring information to the F-14A for the purpose of intercepting the target. The F-14A and the F/A-18 are the only two aircraft with a two-way capability. They are able to respond to a command message with a reply message that consists of status information concerning the F-14A or F/A-18 and its target. The reply message is similar to the command message except that there is no aircraft address code. The E-2C, the only platform that uses the reply, waits for a response at a fixed time delay after transmitting the command message. An E-2C typically will control three or four F-14A fighters at one time. The two-way Link 4A data terminal is known as the AN/ASW-27. Unlike the AN/ASW-25A, it does not have an internal receiver, and operates with an external UHF transceiver.

There are two types of Link 4A message: Control (V) and Reply (R). The Control message is made up of four major parts: sync, address, label and information. The Reply message is identical except it does not have address information The Control message is 70 bits long while the Reply message is 56 bits long. The bit length is 200 microseconds long resulting in a 5 kilobit per second signalling rate; synchronization data is sent at 10 Khz. At the transmitter each message frequency-shift-keys a UHF carrier with a logic one increasing the RF carrier by +20 Khz and a zero decreasing the RF carrier by 20 Khz. The message includes a number of parity bits which are used at the receiver to determine if the message is free from error. If the message contains errors it is rejected. Messages are transmitted at pre-determined update rates depending on what mode of operation the equipment is in. Missed messages are tolerated until they reach a threshold at which point the display equipment in the cockpit indicates loss signal. Link 4A equipments operates in several different modes as was previously mentioned, it also performs several different functions. The original function was and still a prime function is All-Weather Carrier Landing (ACLS) where it assists the pilot in landing the aircraft. The second function is the E-2C to F-14A data link mode where the E-2C vectors the F-14A to a point or a target. A third mode is as a port to update the inertial system. Inertial alignment is usually performed on the carrier deck and once required connecting cables to each aircraft, the addition of the CAINS mode permits the updating of multiple aircraft virtually simultaneously.

The data link procedure for the two-way mode of operation follows the following format. The E-2C normally controls several aircraft simultaneously. As the controller he will transmit a control message to the first F-14A. The called F-14A will recognize his address and decode the message. Normally the control message will trigger a specific reply message which the F-14A will begin to prepare. The two other F-14's will reject the message as it was not addressed to them. The E-2C waits a pre-determined time and then transmits to the second aircraft. the same process will occur, the addressed aircraft receives the message and prepares its reply. After the second control message has been transmitted the aircraft that was called first will begin to transmit his reply. The E-2C knows automatically who the sender is by the time elapsed from the first transmission. This is the reason for the deletion of the address on the reply message. The E-2C then calls the third aircraft afterwhich the second aircraft sends its reply message. After initial start-up time is used efficiently as control and reply messages alternate. See figure 4.

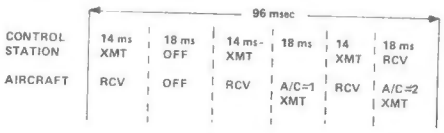


Figure 4. Link 4A Timing

LINK 11/TADIL A

During the 1950's it was recognized that a more precise and real-time method of conveying target and position information among members of the battle group was needed. Radars, inertial and radio naviagation systems were all in development to provide accurate target and navigation information; but little value would be gained, if the combatants continued to use voice as the means of passing this information from one platform to another. To solve this problem the Navy developed the Link 11/TADIL A system.

The Link-11 digital communications link is used to exchange two basic types of information between similarly equipped stations. The two types are:

1) Data messages, including target messages and status/order messages
2) Control messages (to regulate the network)

In addition to these two basic types, net synchronization and net test transmissions are used at the time the network is being established. The network operates on either HF or UHF. HF operation uses single sideband techniques using either upper, lower or both sidebands for frequency diversity. UHF operation uses Frequency Modulation (FM) techniques. Once the network is activated, all operation is unattended and automatic. The HF band was initially selected because it would permit ship-to-ship connectivity over distances much in excess of the radio horizon. The advent (and preponderance) of Link 11-equipped aircraft resulted in the development of the alternate UHF capability.

Since the HF band was the first choice, the waveform selected depended heavily on compatibility with existing signals in the HF band, thus a multi-tone waveform having the same frame length as radio teletype and using multiple audio tones that when summed appear to replicate a voice signal were chosen. This signal is fully compatible with a typical HF transmissions and thus uses a standard HF radio.

The Link 11 waveform is a multi-tone differentially phase-shift keyed signal that has a time period of 13.33 milliseconds. The tone spacing is 110 Hz beginning at a 605 Hz Doppler tone with some tones deleted depending on the mode of operation. At the frame interval, each tone is phase shifted one of four phase values (figure 5).

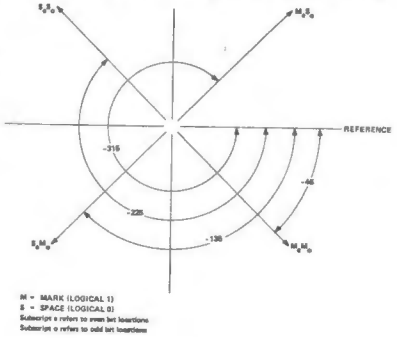


Figure 5. Modulation Vector Diagram

The information is carried in the phase shift from frame to frame. The set of four phases results in two binary digits per tone. As was previously indicated, there are several modes of operation wherein the number of tones are varied. Link 11 usually operates with a data rate of 2250 bits per second (bps) which requires 15 tones. It can also function at a slower rate of 1364 bps with an extended frame length of 22 milliseconds. Recently, the equipment has been developed with a 16 tone or 2400 bps capability which could be used for vocoded voice or any 2400 bps serial bit stream. For transmission, the tones (listed in Table I) are simply summed together and are provided at baseband (nominally 3 KHz) to the input of an HF single sideband radio. In the early days of Link 11, each tone was generated by its own oscillator and the detection process was performed by a bank of analog filters. Both oscillators and filters had a tendency to drift, causing a degradation in performance as the modems aged. They also required frequent and complex maintenance to provide satisfactory performance. Today, digital designs have virtually eliminated the drift problem. The tones are digitally synthesized from a stored replica of a single tone (generally 55Hz), and the use of a bank of digital matched filters in the detection process has eliminated drift as an equipment problem.

TABLE I - LINK 11 TONE LIBRARY

Number	Freq.(hz)	Description	Bit Loc.
	605	Doppler	
1	935	Data	0 and 1
2	1045	Data	2 and 3
3	1155	Data	4 and 5
4	1265	Data	6 and 7
5	1375	Data	8 and 9
6	1485	Data	10 and 11
7	1595	Data	12 and 13
8	1705	Data	14 and 15
9	1815	Data	16 and 17
10	1925	Data	18 and 19
11	2035	Data	20 and 21
12	2145	Data	22 and 23
13	2255	Data	24 and 25
14	2365	Data	26 and 27
15	2475	Data	28 and 29
16	2585	Data	30 and 31
	2915	Sync	

The Link 11 data terminal set performs more than just the modem function; it generates and recognizes both the preamble and control codes that control the data flow in an NTDS network. It also provides the computer with data requests and effectively controls the transmission of information automatically. The message usually consists of a preamble, a start code, data, and a stop code. The preamble frames consist of two audio tones: a synchronization tone, and a doppler pilot tone, generated wholly within the data terminal set. The synchronization tone has a frequency of 2915 Hz, and is phase-shifted 180 degrees each successive frame. With these abrupt phase shifts, the synchronization tone enables rapid frame transition detection and synchronization for the receiving data terminal set. The doppler pilot tone has a frequency of 605 Hz. It is continuous wave (CW) signal that is transmitted at twice the power level of the synchronization tone, with the composite root-mean-square (rms) amplitude of both tones equal to that of the full complement of data tones plus the doppler tone (0 dbm). The doppler tone enables the modem to detect and correct for doppler shifts up to $\pm$75 Hz in the received signal. When control or data frames are sent, the sync tone is dropped and synchronization is maintained by sampling the signal energy in the 825 Hz slot where there is no tone. Following the five frames of preamble, there is a phase reference frame containing all of the data tones. Next are two frames of START code followed by a variable number of data frames followed by two frames of STOP code. See Figure 6. The data bits in the normal mode originate in the computer in blocks of 24 bits. These bits are then encoded by a (30, 24) Hamming code. All data words consist of 24 bits of data and 6 bits of

Hamming code while all control words are 30 bits. The control words are not transferred to the computer but are processed in the data terminal set. The thirty bits are split into fifteen sets of two which develop the one-in-four phase shift. The Hamming code has the ability to detect three errors and correct one error. In practice, the correction of bits is seldom used, because when errors occur in HF transmissions, there are usually more than one error, therefore, the error detection capability is only used to accept or reject the data blocks.

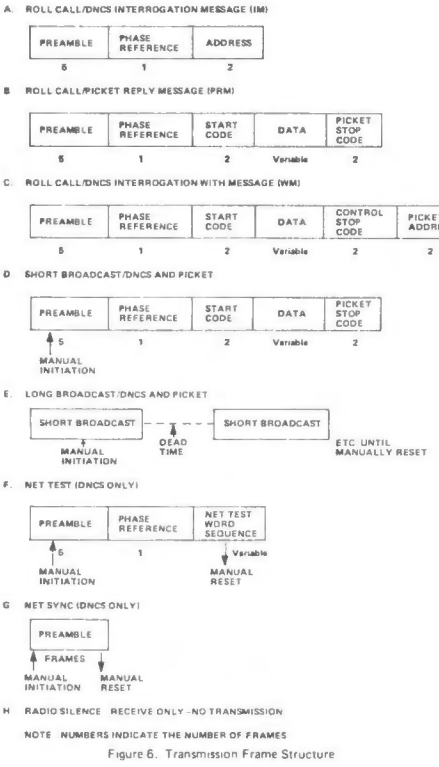


Figure 6. Transmission Frame Structure

Link 11 or TADIL A operates as an automatic netted data communication system. All the participants in the net follow an orderly and prearranged sequence of operation. A common radio frequency is used with each station time-sharing the frequency for data transmission. When not transmitting, each station monitors the frequency for transmissions from other stations. Transmission on the net frequency is restrictied to a single station within any given time. Order in the TADIL A net is maintained by assigning one participant as Net Control Station and all other net units as subordinates (pickets stations). A representive net is shown in Figure 7. The net Control Station is able to maintain order in the net through the use of addresses and control codes. Each participating unit is assigned a unique address code.

Link 11 has four major modes of operation: Roll-Call, Broadcast (long and short) Net test and Net Sync. In the normal mode of operation, one platform is the Net Control station; the rest of the platforms are designated as pickets. The Net Control Station will begin net operation by broadcasting a short message known as an interrogation which may but normally does not include data and which is addressed to a specific picket. The picket platform addressed, as well as all other pickets will hear the transmission. The picket will respond with a message structured according to the description in the previous paragraph and displayed in figure 6. All participants including the Net Control Station will hear the picket reply . When the picket has completed his transmission, the Net Control Station will again transmit an interrogation message only this will be addressed to the next picket in the sequence. When all of the pickets have broadcast their data, the Net Controller will transmit his own data. This completes one cycle of the net. The net cycle automatically repeats until termination. The time required for all pickets to be interrogated is variable and depends upon the number of pickets in the net and the amount of data each transmits. A TADIL A net is capable of operation with at least 20 participating units. In general, the picket stations are pre-set to respond automatically to the Net Controller's call and thus a radio operator is not required. Typical data consist of target position information gained from one of the platform's sensor systems such as its RADAR. Also, periodically the originator supplies his own position as a reference for the target data. The net organization described is known in Link 11 terminology as "ROLL-CALL". An alternate mode of organization known as "ROUND ROBIN" was once used. It both reduced the net cycle time by eliminating the need for a Net Control Station to call each picket and reduced the roll of the Net Contoller by having each picket add the address of the next picket in line. Thus, the Net Controller only started the cycle off and after that it automatically stepped around the network. This mode was unreliable however because in the HF media the participants are weakly connected resulting in situations where the net would hang-up because the addressed picket did not respond causing awaiting pickets to take initiative on their own. There is also a Broadcast mode where a platform will simply broadcast its information. The Net Control Station could do this on a continual basis via the long broadcast, while a picket usually will just broadcast its information once (short broadcast mode). Finally, there are two modes used during Net set-up known as Net Test and Net Sync. The Net test mode allows the network and its participants to verify operation by the transmission of a known test message.

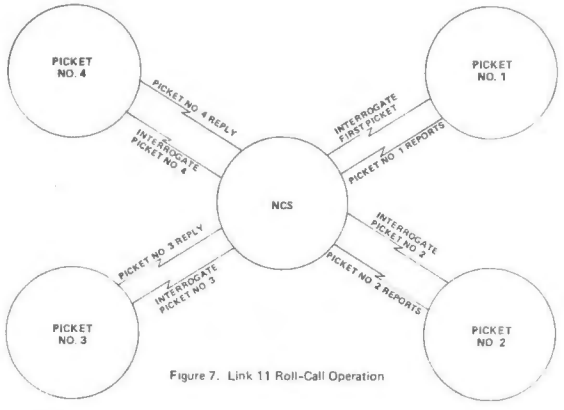


Figure 7. Link 11 Roll-Call Operation

LINK 14

Link 14 is a semi-automatic link which provides computer controlled broadcasts of tactical data by a designated NTDS station for reception by non-NTDS units via conventional radio

teletype transmission (RATT). It operates in either the HF or the UHF frequency band and serves the function of tying non-NTDS ships (generally smaller or older ships) to the overall task force tactical data system. The modulation is the standard frequency-shift-key modulation for teletype. At UHF, a 700 Hz tone indicates a mark is being sent while a 500 Hz tone signifies a space. In HF. a positive and negative shift of 425 Hz about 850 Hz represents a space and mark respectively. A Link 14 broadcast consists of two message types: Status Messages and Track Position Messages. When both status and track position messages are transmitted, this is referred to as a cycle. The duration of the cycle varies accorcding to the number of tracks but is typically 1 minute. This information is broadcast continuously at 1 minute intervals with specific track and status words being updated continuously. The specific track information consists of class, category, position, identity, course, speed, height or depth, and track number. The output of the Link 14 system is a standard 100 words per minute teletype.

THE BASIC PLATFORM SYSTEM

The basic tactical data system is typlified by the equipment on an aircraft such as the E-2C. Shipboard systems tend to be more complex in that they involve many more sensor inputs as well as multiple computers and many more operator terminals. A typical tactical data system consists of four subsystems: the sensors, computers, communications, and displays. The sensor system itself may consist of several different types of sensors some of which are directly connected to the Tactical Data system while others use a manual input for connection. Usually the platform's RADAR and IFF (Identification Friend or Foe) systems are directly connected. This means that information gained by these sensors is directly entered into the computer although some additional inputs may be required of the operator. Other systems may be totally dependent on the operator for entry into the system. The second equipment and the brain of the system is the computer. Each of the three TDS equipped aircraft uses different computers. The computer provides a real-time processing capability of target data and integrates the platforms weapon and sensor systems. The communications equipment provides the mechanism for transmitting and receiving track and position information from other platforms. And finally the Operator Display and Control is the user interface. A display of enemy and friendly units is available on the platform with keyboards and control consoles allowing the operator to enhance the data already in the system.

THE SENSOR SUBSYSTEM

Typical of the sensor subsystem is the platform's radar. A platform entering the coverage area of the RADAR would appear on the RADAR operator's screen. Initially only range and azimuth information would be available but as time goes on additional information such as speed and altitude may be determined. The RADAR operator may elect to transmit an IFF interrogation which would further classify the track as a friend or foe. Additional sensor inputs may further resolve the nature or intention of the track. All of this information is stored in the computer.

THE TDS COMPUTER

The computer keeps files on all tracks received from its own sensor subsystems. As information is accumulated the files are updated. The computer also receives position information of itself from its navigation computer. A number of software routines are available that discriminate between inputs from more than one sensor, derive additional data (for example velocity), maintains track files and in general assist the operator. The computer operates on a twenty-four bit word which are usually received in pairs. It is able to process 75 words in a second. Typical of airborne computers are the Litton L-304 used on the E-2C, the Univac CP-901 used on the P-3C and the AN/UYK-10 used on the S-3A. In the past, the choice or selection of the computer was the perogative of the airframe manufacturer and has been regarded as Contractor furnished equipment. Typical of shipboard computers are the AN/UYK-7 and the CP-642A/B. The software used on these computers undergoes a continual process of upgrade and modification at Fleet computer support centers.

THE COMMUNICATIONS EQUIPMENT

The communications subsystem consists of an antenna, sometimes an antenna coupler, a radio, a modem and their associated controls (figure 8). The two frequency bands used in the Fleet today are the Navy's UHF and the HF. The choice of frequency depends on whether or not the platforms are within line-of-sight range or not. Link 11 and Link 14 are capable of operation on both UHF or HF channels while Link 4A is used strictly in the UHF band. The baseband computer data is confined to bandwidths commonly used in these frequency bands so that once the data has been modulated and up-converted to the appropriate RF frequency it makes use of the frequency channel in much the same way as typical voice users. The radios themselves are generally capable of tuning to any one of the frequencies available in the band (280,000 in HF and 7000 in UHF) but from a practical frequency management viewpoint will tend to operate on a small set of prearranged frequencies. The Modem, which stands for modulator-demodulator, converts the digital data from the computer into an analog baseband signal which is suitable for transmission in the appropriate channel. In the receive mode it performs the reciprocal function. It has the added responsibility of generating some of the link protocols and will actually control the operation of the link.

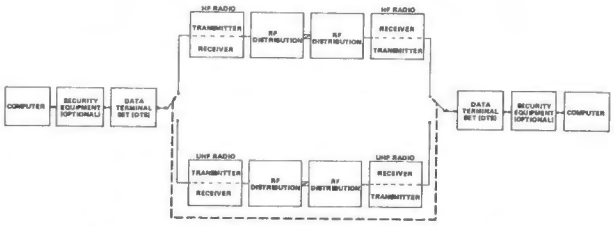


FIGURE 8 - A Typical ATDS system

DISPLAY CONSOLE AND INPUT/OUTPUT

The Display console is the man/machine interface with the system. It displays the data resident in the computer and allows the operator to manually input amplifying information. In some cases it can be used to cause action as the operator can cause certain messages to be transmitted to others. The actual display takes two forms: one is a Cathode Ray tube similar to the RADAR screen which displays various symbols representing ships, aircraft and submarines and whether or not they are friendly, enemy or unknown. Figure 9 displays some of the basic NTDS symbols. The second display varies from console to console and platform to platform. On aircraft it is frequently in the form of a Menu allowing the operator to choose among modes or to enter input data . Some consoles have multiple readouts and bottons for doing this. Figure 10 is typical of the NTDS display console. Operators at these consoles, by button actions and other controls, communicate data and orders to, and receive processed information and recommendations from, the computer program. They detect, enter, and update target received from ownship sensors and non-TDS units, control assigned aircraft, maintain geographic and relative navigational positioning, assign targets to, and receive status reports from ownship weaponry, evaluate tactical situations, and exchange status and order information between units. The most significant capability of an NTDS-equipped task force, compared with a conventional configured force, is its ability to continously provide all NTDS units with a complete and accurate display of the force tactical situation, instantaneously, as it progresses. This information, displayed at all NTDS consoles, consists not only of complete target data, but also up-to-the minute engagement status for all NTDS force weapons systems.

SYMBOL	MEANING
	SURFACE FRIENDLY TRACK
	AIR FRIENDLY TRACK
	SUBSURFACE FRIENDLY TRACK
	SURFACE UNKNOWN TRACK
	AIR UNKNOWN TRACK
	SUBSURFACE UNKNOWN TRACK
	SURFACE HOSTILE TRACK
	AIR HOSTILE TRACK
	SUBSURFACE HOSTILE TRACK
	OWNSHIP
	AIRCRAFT CARRIER FRIENDLY
	SURFACE HOSTILE ENGAGED
	SUBSURFACE HOSTILE ENGAGED
	SURFACE FRIENDLY ENGAGED
	CAP STATION
	CORRIDOR

Figure 9. NTDS Symbology

TYPICAL NTDS DISPLAY CONSOLE

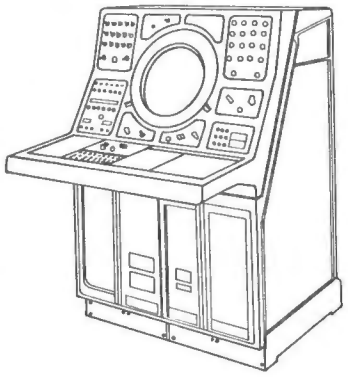


FIGURE 10

REFERENCES

Joint Chiefs of Staff Publication No. 10, "Tactical Command and Control, and Communications Systems Standards", Available from Joint Chiefs of Staff, ATTN: Documents Division, Wash. D.C. 20301.

North Atlantic Treaty Organization Standardization Agreement (STANAG) No. 5511, "Tactical Data Exchange - Link 11".

MIL-STD-188-203-1, Military Standard for Subsystem Design and Engineering Standards for Tactical Digital Information Link (TADIL) A, dated 10 Sept. 1982.

MIL-STD-188-203-2, Military Standard for Subsystem Design and Engineering Standards for Tactical Digital Information Link (TADIL) B.

MIL-STD-188-203-3, Military Standard for Subsystem Design and Engineering Standards for Tactical Digital Information Link (TADIL) C.

Rockwell International Introductory Description entitled "Link 11 Network Operation with Collins Data Terminal Set AN/USQ-59"

Schoppe, W., "The Navy's Use of Digital Radio", IEEE Trans on Communications, Vol. Com-27, No. 12, December 1979.

IMPLICATIONS OF JOINT TACTICAL INFORMATION DISTRIBUTION SYSTEM/TACTICAL DIGITAL INFORMATION LINK J ON NAVY AIRCRAFT WEAPONS SYSTEMS

Roy W. Kuhn

Unit Chief - Electronics
McDonnell Aircraft Company
McDonnell Douglas Corporation
St. Louis, Missouri

ABSTRACT

JTIDS/TADIL J is an advanced communications system providing a data and voice transmission capability via digital means. It will have far reaching implications on future Navy command, control, and communications applications. This paper addresses the tasks to be considered when incorporating the system into a Navy tactical aircraft. Tens of thousands of data bits per second can be exchanged by fighter/attack aircraft with other JTIDS equipped aircraft, ships, and land units. This information includes location of friendly, unknown and hostile air, sea and land elements, as well as command and control information. Effective use of this vast source of information impacts the aircraft data integration, and controls and displays. Significant additional onboard processing speed and program memory will be necessary to sort, filter, and store the desired data. It must then be merged with information from onboard sensors. New and revised display formats will be required to present the data to the aircrew in a timely and rapidly comprehensible manner for maximum pilot operability. The incorporation of JTIDS and the full TADIL J message standard by the Navy will be accomplished and operationally absorbed in stages.

INTRODUCTION

Radios provide communications. The military has used radios for command, control, and mission coordination ever since Lt. H. A. Dargue and Lt. J. Maeuborgne of the U. S. Army demonstrated 2-way air-ground communications at Manila in the Philippines in December, 1914(1). Early installations were simple, requiring only electrical power, an antenna, the radio itself, a microphone or telegraph key, and a speaker. The aircraft usually carried a radio operator who, in addition to operating the radio, also carried spare parts for repair of malfunctions. The operator provided decoding of the information (from voice or dots/dashes) and storage for future reference (in his mind or on paper). Modern UHF/VHF radios in fighter/attack aircraft have these same requirements with the pilot as the operator.

Since 1914 communication countermeasures and jamming have progressed to the point where, today, the enemy can essentially preclude conventional voice communications on the battlefield in nearly all frequency bands. This has required the development of jam resistant communications systems.

JTIDS

The Joint Tactical Information Distribution System (JTIDS) is an advanced form of radio system which goes well beyond the traditional radio and earlier data link systems. It will be utilized by a large community of surface (land and sea) and airborne platforms (Figure 1). Through the use of psuedo-randomly selected transmit frequency selection, time separation, and phase code hopping of the data bits, it provides secure message capability with excellent jamming resistance in a high ECM environment.

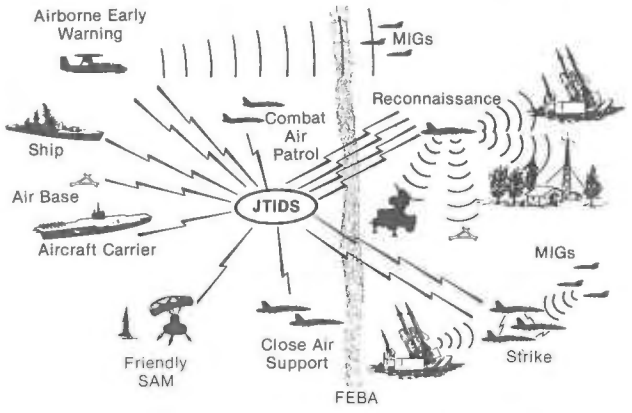


Fig. 1 JTIDS/TADIL J Community

JTIDS provides a verbal communications capability; it does this by digital means. More important, however, by effective implementation, JTIDS gives a digital representation of tactical data between participants. Up to 300 thousand bits per second (KBPS) of data may be exchanged between members of a battle group.

JTIDS also provides highly accurate relative navigation capability between platforms. By the use of "round trip timing" (RTT) messages, the receiving JTIDS "terminal" (the radio hardware in an aircraft or ship) in one platform advises the transmitting terminal in another platform of the time that its messages arrive. Thus, based on the propagation time (about one foot per nanosecond), the transmitting terminal computes the distance to the receiving terminal. Triangulation then provides precise relative location.

TADIL J

The Tactical Digital Information Link J (TADIL J) is a total information link being developed to implement a joint interface among all the U. S. services and NATO countries. TADIL J is that

portion of the JTIDS interface design that prescribes the message rules, procedures, format, content, and minimum implementation requirements for units participating on JTIDS. TADIL J specifies that "fixed format messages" will be transmitted via a series of 75 bit data words with each bit in each word of each message containing predefined (fixed) information. TADIL J also defines required responses when these messages are received.

STAGED INCORPORATION

Implementing an interface of the magnitude and complexity of JTIDS/TADIL J requires considerable resources on the part of the services. The U. S. Navy is accomplishing its implementation in stages. The first stage will be the development and testing of the JTIDS hardware and its installation in a few of the platforms on which it ultimately will be installed. The development and implementation of the TADIL J message standard and its protocols and incorporation of the JTIDS equipment into all other intended platforms will be accomplished in follow-on stages.

The F/A-18A will be the first U. S. Navy fighter or attack aircraft weapon system to incorporate JTIDS. The system will be installed in two test aircraft early in 1986 and delivered to the Navy for flight test in May 1986. This is the culmination of several years of study at McDonnell Aircraft Company (MCAIR) regarding the impact of installing JTIDS into the F/A-18A. Later in this first implementation stage, one Navy aircraft carrier, its E-2C Hawkeye aircraft, and 24 F/A-18A Hornets will be equipped with JTIDS. They will be used to certify that JTIDS does indeed provide the tactical, multi-net, automatic, secure, jam resistant communications for which it has been designed.

The development and integration of TADIL J is a major part of the second stage. It will ultimately see JTIDS/TADIL J widely incorporated in ships, aircraft, and ground stations in the 1990's. Stage II is in an early concept design definition phase.

The implementation of JTIDS/TADIL J will have a far reaching impact on Navy operations. The implications of its incorporation on fighter and attack aircraft operations is discussed in Reference (2).

JTIDS IMPLICATIONS

The task of incorporating a major system such as JTIDS into a tactical aircraft requires integration so that it will correctly supply its data and not interfere with existing equipment. Proper operation depends on sufficient data processing, integrated controls and displays, physical installation, cooling air, and interconnect wiring. JTIDS imposes a considerable impact on the aircraft's available resources.

Data Processing. When developing the specifications for its JTIDS system, the U. S. Navy determined that the terminals for all non-command and control aircraft would be identical. The same terminal can be installed in an F/A-18A, EA-6B or F-14D. This precludes having the terminal perform any processing based on the particular type of host vehicle in which it is installed.

JTIDS terminals communicate with each other via a highly sophisticated series of digital pulses. A large amount of initialization data must be provided before the terminal will attempt to communicate these pulses with other terminals. Much of this initialization data is changed on a daily basis for security purposes. These pulses form the 75 bit data words exchanged by terminals.

After reception and decoding, the terminal outputs these 75 bit words via a series of 16 bit words over a MIL-STD-1553B digital multiplex (MUX) data bus to the host platform's computer. There it is examined to filter, select, and store the desired information.

Installation. Terminals for fighter/attack type aircraft (Figure 2) are comprised of two units; a Receiver Transmitter (R/T) and a Data Processor (DP), weighing 40 and 60 pounds respectively. A Secure Data Unit (SDU) is attached to the DP to provide security codes. The units dissipate a total of 1950 Watts of heat and require cooling air from the aircraft's Environmental Control System. Because of the need to enter new security codes on the SDU on a daily basis, the DP must be easily accessible.

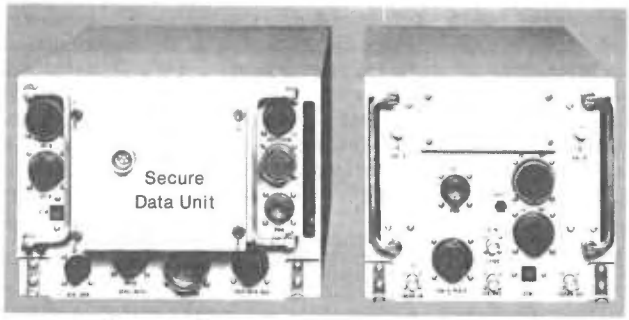


Fig. 2 JTIDS Class II Terminal

These "Class II" terminals give the tactical aircraft pilot two channels for voice communications. Other "classes" of terminals (e.g., those in command and control units) provide from 4 to 10 voice channels.

Wiring. The interface of the JTIDS terminals to their host platforms consists of primary power, antenna coaxial cables, MIL-STD-1553B MUX busses, microphone wires, and blanking signals. The microphone and MUX bus wiring present unique considerations since they carry classified data. These must be handled so that the information they carry cannot be detected outside the aircraft.

The antenna-to-antenna coupling of JTIDS signals to other systems on the platform may cause interference between the systems unless the proper blanking signals are used.

TADIL J IMPLICATIONS

To satisfy all of the planned JTIDS users, a message standard with many types of messages is being developed. Figure 3 illustrates the functional areas covered by TADIL J.

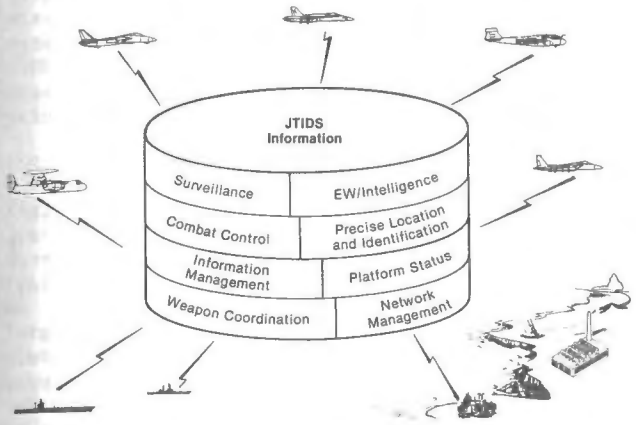


Fig. 3 TADIL J Functional Messages

In the early stages, the goal is to verify that required communications can be achieved. To reduce risk, cost, and complexity and to focus resources on the JTIDS equipment development, only a limited number of message types will be exchanged. Terminal communications messages, existing U. S. Navy Link 4A Air Intercept and Control (AIC) messages, and the Precise Participant Location and Identification (PPLI) messages of TADIL J will be exchanged by aircraft. The full capability of TADIL J will be implemented later.

RECEIVED DATA

All terminals receive network management messages to maintain information exchange. However, receipt of messages such as Surveillance (data reported by command and control platforms on friendly, unknown and hostile tracks), PPLI (data each JTIDS equipped platform reports on itself), Targets (data from local flight members and controlling aircraft), and Status (amplifying data on each of the above) will vary as a function of the platforms assigned mission. For fighter/attack aircraft, the amount and distribution of data that might be required to be received and stored is:

Message	Received Tracks	Stored Tracks
Surveillance	1000	150
PPLI	300	200
Targets	100	100
Status	1400	450

In addition, messages supplying Threat Warnings, Electronic Warfare, and Aircraft Control will be received. These all add to the amount of data stored in the aircraft computer. Virtually all of the received data is subsequently processed for weapon system integration and potential display to the aircrew. The amount of data displayed would be a function of the aircraft mission and the criteria initialized by the pilot.

TRANSMITTED DATA

The JTIDS terminal uses the RTT messages to perform the calculations necessary to determine the platform's position. The terminal then formats this information for transmission in the PPLI message. The platform determines its own status (fuel, weapons, equipment, radio frequency) and sends this information to the terminal for inclusion in the PPLI message.

The platform also supplies information to the terminal on targets it is tracking for transmission to the controller and other flight members. This can include position, airspeed, course, and target type on all radar targets. Knowledge on other targets detected by sensors such as infrared detectors, radar emission detectors, and optical devices will also be supplied when available.

INTEGRATION IMPLICATIONS

Integration is the blending of individual items to form a whole. It is also the incorporation of an item into a larger unit to provide a functional system. As shown in Figure 4, JTIDS, TADIL J, and existing aircraft weapon systems are individual items. These need to be blended with the host aircraft to form a whole. However, the aircraft, with these systems completely integrated, also needs to be incorporated into the entire JTIDS/TADIL J community of ships, aircraft, and land units. Without JTIDS/TADIL J on the aircraft, it cannot receive the data they supply. But without the others in the community, there would neither be data to receive nor messages to send.

The implications of integrating JTIDS and TADIL J into an aircraft were outlined earlier. How these can be resolved can best be shown by example; in this case, the F/A-18A.

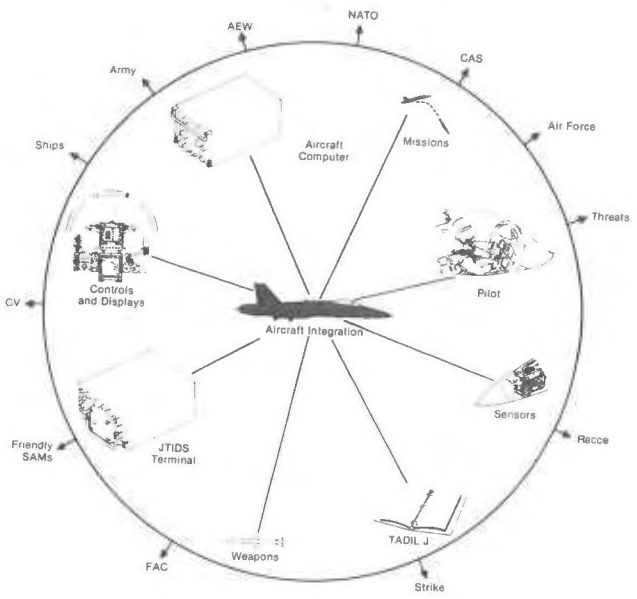


Fig. 4 Aircraft/JTIDS Integration

BACKGROUND

The F/A-18A provides an excellent vehicle for its role in JTIDS development because it will perform fighter, attack, and reconnaissance missions.

Incorporating JTIDS/TADIL J into the F/A-18A is a significant task. JTIDS, while providing a quantum increase in capability, is larger and heavier than the combined total of all the Communication, Navigation, and Identification (CNI) systems already on the aircraft.

Since 1978, MCAIR has performed a series of studies for the integration of JTIDS into the F/A-18A. These were phased to parallel the development of the terminals by the Navy. In 1982 and 1984 there were reviews by U.S. Navy pilots in the MCAIR Manned Aircrew Simulator to examine proposed control and display implementation for JTIDS/TADIL J. These integration studies, complete with manned simulation and pilot reviews, provide the basis for the following discussion.

CLASS II TERMINAL

Since the JTIDS terminal performs the AN/ARN-118 TACAN function which permits its deletion when JTIDS equipment is installed, installation of the R/T in that location was a first consideration. Although the R/T is somewhat larger than the TACAN equipment, sufficient space is gained when the TACAN shockmount is removed as JTIDS is not shockmounted.

Installation of the DP is more difficult because it does not replace an existing unit and because the SDU is located on its front. In addition, the DP should be located in close proximity to the R/T to keep certain interconnecting wires short. Next to the TACAN is the Command Launch Computer for the High speed Anti-Radiation Missile (HARM CLC). The HARM CLC and the TACAN are located behind the quick access door of Bay 3 Right (Figure 5). The DP will be installed in the location presently occupied by the HARM CLC. The DP, like the R/T, is somewhat larger than the replaced unit, but again sufficient space is gained by the removal of a shockmount. The HARM CLC will be installed in a newly developed equipment bay.

The R/T has connectors for two antennas as well as inputs and outputs for blanking purposes. (Signals to the R/T to turn off its receivers when other potentially interfering systems on the aircraft are to transmit, and signals from the R/T to other systems to turn off their receivers when the JTIDS is to transmit). JTIDS will use the same two antennas as TACAN. An antenna-to-antenna compatibility analysis (ATACAP) was performed by MCAIR. It indicated that JTIDS transmit blanking is required to both the AN/ALR-67 Countermeasures Receiving Set low band receiver and the HARM CLC low band receiver. However, if the JTIDS transmit duty cycle exceeds certain limits, then other means (other than via normal blanking signals external to the systems) are needed to reduce interference to acceptable levels. No blanking is required by JTIDS from other transmitters. Final determination of blanking needs will be part of the JTIDS flight test evaluation.

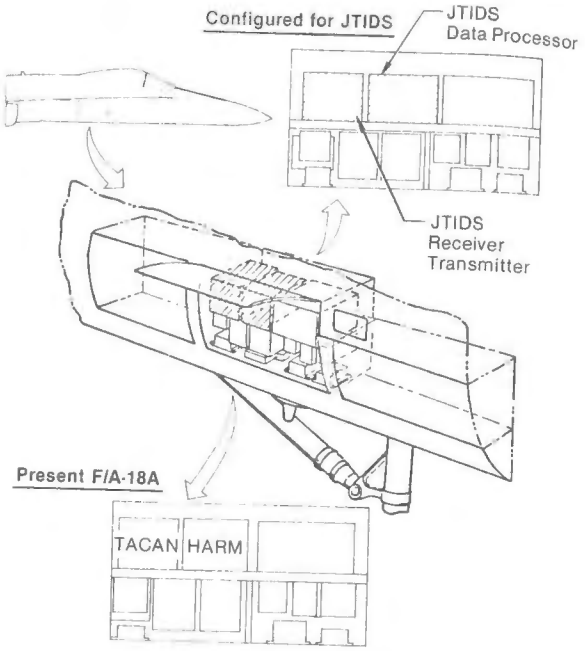


Fig. 5 JTIDS Installation in F/A-18A

AIRCRAFT COMPUTER

The JTIDS terminal provides the received data to the aircraft computer in 16 bit digital data words over the MIL-STD-1553B MUX bus. This consists of all information that the terminal is receiving from the circuits on which it has been initialized to participate. Circuits are the protocols which determine the type, amount, timeliness and update rate of information to be exchanged, who the participants will be and their need to provide and receive the data. This data is filtered by the aircraft mission computers to eliminate that which is unnecessary. As shown in Figure 6, the received data passes through the Platform Information Selectivity filter (filters by message type and data in the message) based on aircraft type (e.g. F/A-18A). The computer then performs Unique Weapon System Tailoring based on the aircraft's present mission (e.g. strike, combat air patrol or escort) and stores it in an onboard database. This tailoring function also filters and sets

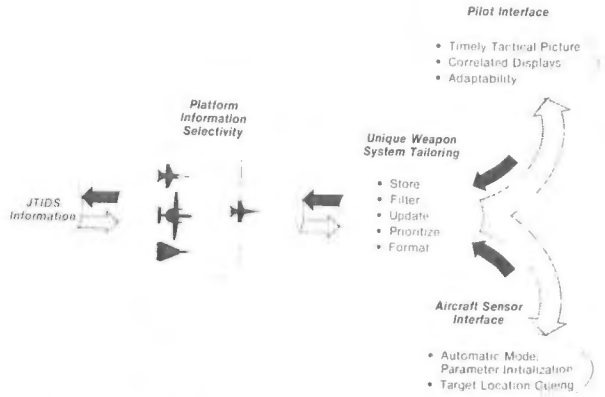


Fig. 6 Aircraft Computer Tasks

priorities to allow display of only that data the pilot wants at a particular phase of a given mission.

The computer determines which onboard weapons, sensors, and displays require the data. For example, if target assignment communications were received from a controller, the information on the target can be provided on a head down display for visual comparison with onboard radar tracks. The data can also be shown on the head-up display (HUD) to cue the pilot and can be furnished to a missile to provide initial targeting information prior to launch.

The aircraft computers provide the terminal with physical parameters such as speed, altitude, heading, latitude, and longitude. The computers also gather information from onboard systems for transmission. All radar and Forward Looking Infrared (FLIR) detected targets will be provided as well as data from future systems (such as Automatic Target Recognition, Infrared Search and Track, or Global Positioning). It will store information required for initialization and send it to the terminal. This must be accomplished before the terminal can attempt communications with other terminals.

Data Busses. The F/A-18A computers (two AN/AYK-14's) communicate with peripheral systems via MUX data busses (Figure 7). These computers are equipped to communicate on three MUX channels designed in accordance with MIL-STD-1553A. Each channel consists of two redundant 2-wire busses (Bus X and Bus Y). The computers communicate with the peripherals over Bus X until a fault in the interface is determined, at which time it will attempt to communicate over Bus Y. The F/A-18A has allocated the interface of the various systems to Channels 1 or 2, to limit wire length, provide functional redundancy, and balance data traffic. The third MUX channel is used to determine which computer is the controller for Channels 1 and 2.

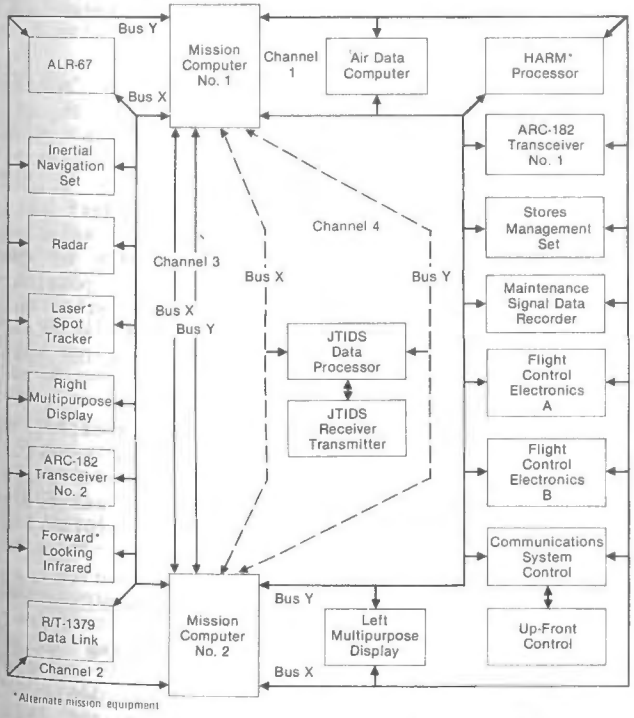


Fig. 7 F/A-18A Avionics Multiplex Data Bus System

There are three new MUX bus problems. First, since the JTIDS terminal communicates via a MIL-STD-1553B data bus, the computer must be redesigned to accept the new protocols (mode commands with single data word transfers, controller broadcast modes, and relaxation of remote terminal response time from 7 microseconds to 12 microseconds). Second, to maintain the security of JTIDS information requires that the decoded data be provided on a MUX bus which interfaces only with systems that are designed to provide a high level of electromagnetic isolation. This prevents the data from being inadvertently radiated outside the aircraft. Finally, the present MUX busses are already heavily loaded, and it would be difficult to add the JTIDS data to them. For these reasons, an additional MUX bus is required and will be used by only those systems which handle classified data.

Processing Speed. JTIDS adds significantly to the processing speed requirements of the aircraft computers. The present AN/AYK-14's each has a throughput capacity of 450 thousand operations per second (KOPS), for a total of 900 KOPS on the F/A-18A. During earlier studies, it was determined that JTIDS requires an additional throughput of about 200 KOPS when initially incorporated and about 725 KOPS when fully implemented. These go well beyond the available reserve on the aircraft. They were based on the processor performing five major functions for JTIDS:

- o Terminal Control and Net Management
- o Relative Navigation/PPLI
- o Tactical Data Integration
- o Controls and Displays
- o Support Functions

Memory. While the implementation of JTIDS caused the processing speed capability of the AN/AYK-14's to be exceeded, such is not the case with it's program memory. For the five functions listed above, plus a database, about 28,000 words of program memory are required for the initial implementation. For full implementation, this grows to a total of about 59,000 words. This is a sizable programming task. It will equal about 39% of the total mission computer program for the entire F/A-18A weapon system incorporated todate (about 150,000 words). The mission computer total available program memory is 256,000 words (128,000 words per computer).

Improved AN/AYK-14. To support increases required for JTIDS/TADIL J and other future systems, the Navy is redesigning the AN/AYK-14 computer. Improved features of the new model will be:

- o 5 MUX channels - each compatible with 1553A/B
- o 128K core memory
- o 128K Electrically Eraseable PROM memory
- o 64K RAM memory
- o Dual single card processors, each with twice the speed of the present AN/AYK-14
- o New special purpose instructions and interrupts

CONTROLS AND DISPLAYS

To understand how JTIDS data may be used, a brief description of the F/A-18A controls and displays

is included here. The current production F/A-18A cockpit is equipped with a head-up display and three multipurpose cathode ray tube (CRT) displays (Figure 8). Each is driven by the aircraft's mission computers. In the early development of the F/A-18A's controls and displays, an important philosophy was established: none of the major display units are dedicated to a particular system (e.g., there is no "radar" scope). However, through pilot usage and experience, the right Multipurpose Display (MPD) is normally employed as the control/display for radar information with the left MPD normally being the control/display for electro-optic and infrared sensors, armament information, and aircraft caution, advisory and built-in-test functions. Under computer control, the pilot can select either MPD to perform any of these functions.

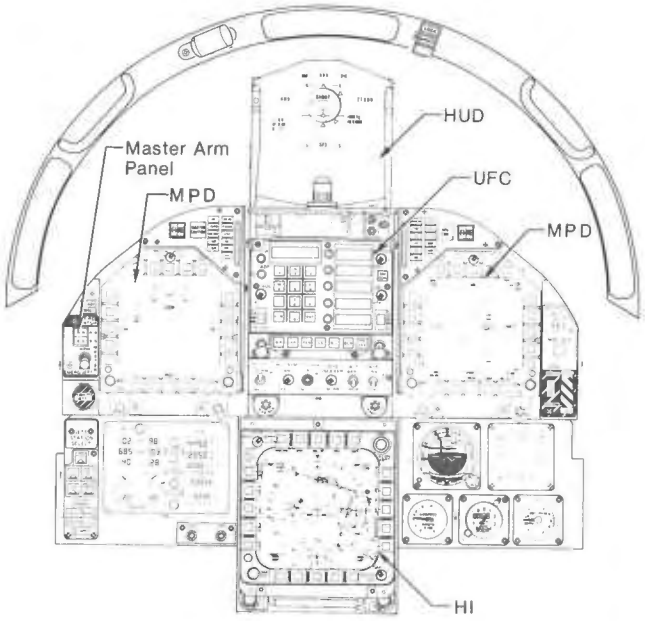


Fig. 8. F/A-18A Cockpit

A second philosophy was also followed. The data displayed is initially dependent on the aircraft master mode, selected by the pilot on the Master Arm panel (Figure 8). When air-to-air (A/A), air-to-ground (A/G) or navigation is selected, each of the cockpit displays presents a predetermined format. (Decisions were automated in the avionics system with manual pilot override when appropriate.) In addition, regardless of the previous mode, activation of only one switch on the control stick (Figure 9), the A/A weapons select, places the entire weapon system in the air-to-air mode. Radar parameters (such as antenna scan limits, frequency, and PRF), steering information on the HUD and initial weapon parameters, are optimized for the weapon selected (forward for Sparrow, down for Sidewinder, aft for gun).

Determining the operation and developing the desired implementations took considerable resources on the part of the Navy and MCAIR. All decisions on how the displays should change for various modes were reviewed by Navy and Marine pilots in the Manned Aircrew Simulator during the aircraft's development in a formalized process

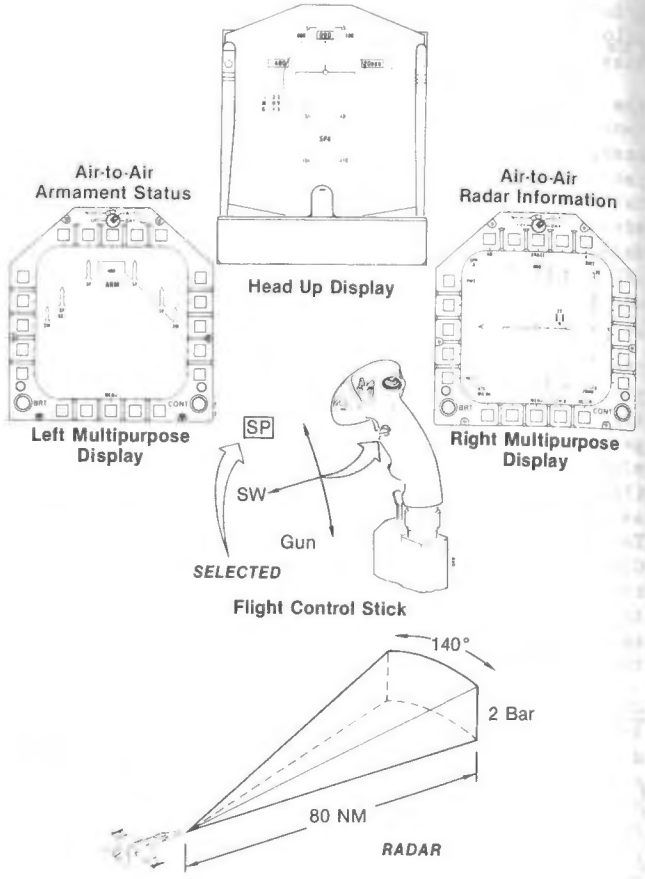


Fig. 9 Automatic System Initialization When Air-to-Air AIM-7 Mode Selected

called "Aircrew System Advisory Panel (ASAP) reviews." Thirteen such reviews were held during development of the Hornet. This approach continues as additional growth capabilities are integrated into the weapon system.

Since JTIDS is being looked upon as a "multi-spectrum" sensor providing data useful in all aircraft master modes, a modification in these philosophies is developing. Data from multiple sources/sensors will be assimilated on the "JTIDS" display. No longer will the left MPD revert to a weapons display when A/A or A/G modes are selected (Figure 9). This display will now remain in the mode previously selected by the pilot; possibly JTIDS. Thus, during an A/G mission, the pilot can maintain an awareness of airborne threats provided by the JTIDS system in addition to the A/G information and vice versa.

JTIDS/TADIL J Display. During prior studies, a method for displaying the tactical data supplied by JTIDS/TADIL J evolved (and is still evolving). With digital computers and programmable multipurpose displays, displayed data can be "information of the moment." As new information becomes available through the continued evolution of JTIDS/TADIL J, the format of the display can be revised without changing the display hardware. For example, the format for displaying JTIDS Tactical data (J-TACT) shown in Figure 10 is the result of a review by a group of Navy pilots in April, 1984. Using the pushbuttons on the left

side permits the pilot to select the desired data. It provides a method of displaying all data that will be available during the initial JTIDS implementation stage.

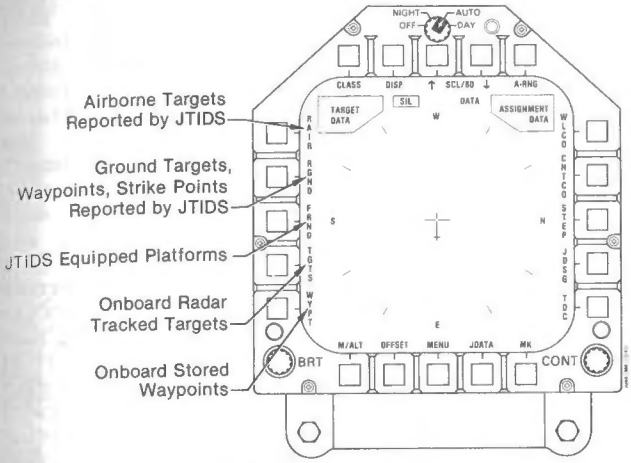


Fig. 10 Initial J-Tactical Display

Pilot selection of options provides for the merging of data which reduces his workload while improving combat effectiveness. For example; by selecting the TGTS option, the pilot requests that onboard radar detected targets be overlayed on the JTIDS display for comparison to JTIDS tracks. In the future, as other new tracking systems are incorporated, selection of their data for display to the pilot will be made available. Thus one display may be used for visual correlation of the data from multiple onboard sensors and from other platforms.

In later stages, the available pilot options on the left side of the display will be changed to incorporate greater selection of data filtering and prioritizing to accommodate new messages, such as:

- o PPLI (Subsurface, Ground Tracks)
- o Surveillance (Air, Surface, Subsurface, Ground, Electronic)
- o Electronic Warfare/Intelligence
- o Weapons Coordination and Management
- o Control (Tactical, Air, Traffic, Track)
- o Threat Warning
- o Intra-Flight Exchange of Targets/Commands

A possible ordering of the selections which can be provided to display this additional data is shown in Figure 11. It utilizes submodes, or levels of indenture, to give the pilot greater selection capability. For example, selecting RAIR would allow the pilot to select from four or five submodes for the display of hostile or unknown air (HUAIR) targets, or friendly air (FAIR) targets, to preclude displaying more targets than the pilot can effectively discriminate between.

Horizontal Indicator. Overlaying JTIDS derived information on a ground map could be of great benefit. Waypoints, strikepoints, and the location of surface-to-air missiles (SAM's) could be viewed with relation to the terrain. The F/A-18A Horizontal Indicator (HI) provides this capability. As

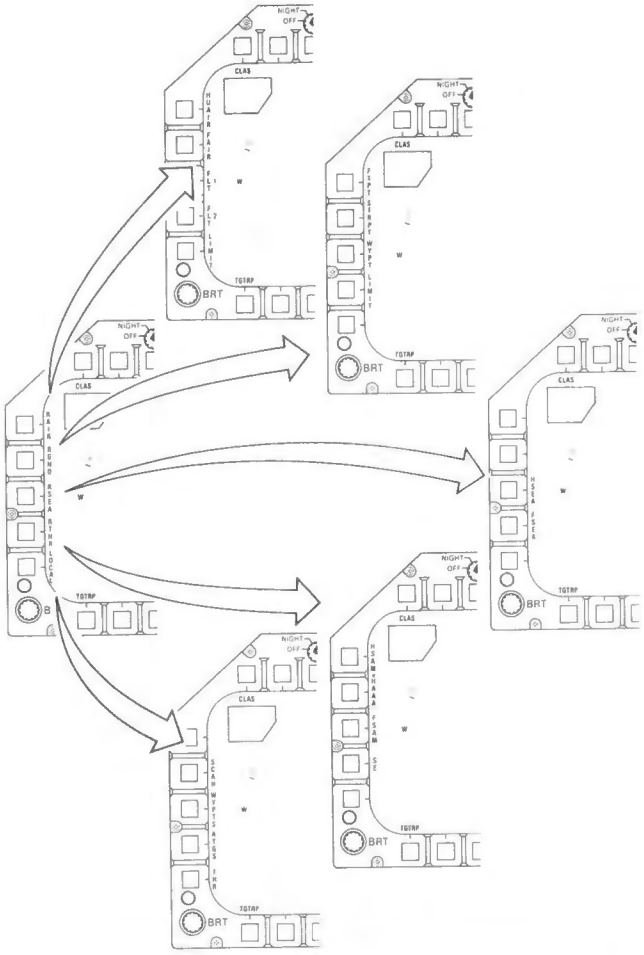


Fig. 11 Future Potential J-TACTICAL Display Submodes

shown in Figure 12, when the location of a particular type of SAM is provided by JTIDS, the lethal envelope of the SAM will be computed for the planned attack altitude or the aircraft's current

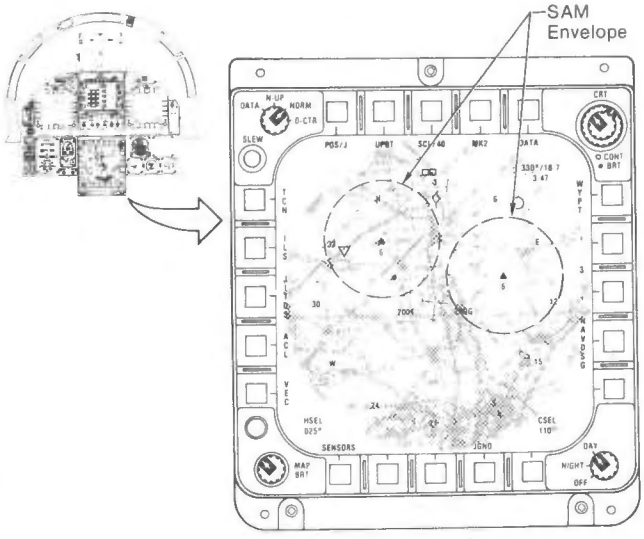


Fig. 12 Horizontal Indicator

altitude using prestored data on each SAM type. The pilot can then compare this area with the terrain to determine minimum risk areas due to terrain masking.

Radar Display. While the J-TACT display format provides for the merging of data from several sensors, there may be occasions when it is desirable to use the left MPD for other purposes. At these times, the pilot may still desire to correlate visually JTIDS data with onboard radar data. By deselecting Reject (REJ) on the radar display (Figure 13), the four highest priority JTIDS targets will be overlayed on the radar display.

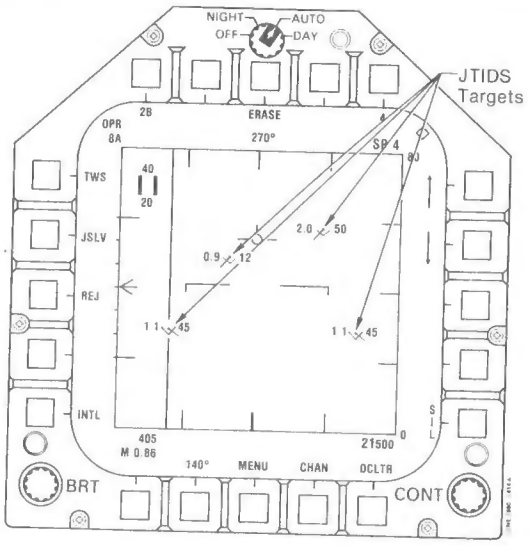


Fig. 13 JTIDS Data on Radar Display

The pilot presently can perform manual lock-on of radar detected targets. It's also possible that the pilot will want to attack JTIDS targets not yet detected by the radar. This presents a difficult task that onboard automation can greatly simplify. Since JTIDS target data is transmitted with great accuracy, this data will be used for initial radar pointing. When the pilot has determined that he wishes to attack a particular target provided on the JTIDS net, he designates that target on the J-TACT display (Figure 14). The JTIDS Slave (JSLV) option appears on the radar display. When the JSLV option is selected, the radar antenna is pointed at the JTIDS provided target location and the radar acquisition gate is positioned at the proper range to optimize acquiring the desired target. The radar is allowed several seconds to lock-on. If lock-on is not achieved within the alloted time, the radar reverts to its previous mode.

Voice Control. Like many current Navy aircraft, the F/A-18A provides the aircrew with two radios for audio communications. Normally, each is tuned to a different frequency. The two JTIDS voice channels will be implemented in a similar manner in the Hornet. Each channel may be set to operate on one of many available JTIDS circuits.

After JTIDS voice information is received, it requires special handling to prevent the emanation of its classified information. In the F/A-18A,

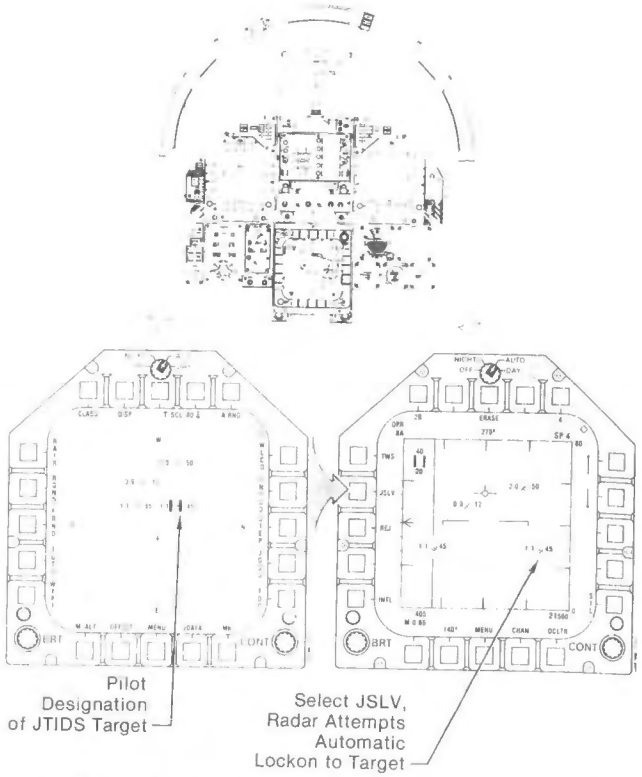


Fig. 14 Target Designation Provides the Pilot With JTIDS SLAVE (JSLV) Option

the AN/ARC-182 radios already provide secure voice capability through the KY-58/TSEC Secure Speech Encoder. These secure audio circuits are isolated from other nonsecure circuits in the Intercommunication Set (ICS). However, redesigning the ICS will be required since simultaneous reception on two secure circuits will now be possible.

Terminal Control. The JTIDS terminal requires a "predeployment" load and a "premission" load. These parameters enable it to communicate with other terminals. The amount of information required in these loads has been estimated at several hundred to thousands of words of data. Manual entry by the pilot or a ground crew member in the cockpit would be exceedingly tedious, time consuming, and subject to error. Therefore, JTIDS/TADIL J requires a device that will allow the pilot to perform the loading task prior to entering the cockpit. In addition to the initialization data, other mission planning information such as waypoints enroute, strike points, and target parameters may be included. The data would be loaded into a Data Storage Unit (DSU) and carried by the pilot to his aircraft. The DSU is anticipated to be approximately the size of an eight track tape cartridge and weigh two pounds.

JTIDS crypto function controls (HOLD and ZEROIZE) are required to load and maintain the crypto variables. These variables are provided from a crypto loading device (hand carried to the aircraft by the crypto officer) to the SDU on the DP. Since the APX-100 IFF system is integrated into the F/A-18A with a zeroize switch on the ICS, these JTIDS crypto functions also will be added to the ICS.

The terminal may be preloaded with parameters for operation on up to 128 "metachannels." A metachannel represents the totality of data exchanged between participants on a net. It provides over 300,000 bits per second of data rate capacity. Each metachannel can have several types of circuits implemented therein. Presently, the terminal in fighter/attack aircraft can operate on circuits in up to six of the 128 metachannels simultaneously. These are not changed in flight. The display formats for control of these metachannels are being developed. However, Navy operations may require that others be selected from the 128 initialized as different phases of the mission are entered. The F/A-18A MPD's, in conjunction with mission computer software control, can provide this required capability if it should be determined advantageous during flight test operations early in the development program.

Color Display. A consideration not previously discussed, and yet to be defined, is the level of information the aircrew can effectively handle. Whereas the display of 20 or 30 targets would probably not present any problem to the controller in a C^2 aircraft who does not have to concern himself with the task of flying the aircraft, this may equal or exceed the capability of a combat pilot. The use of color to differentiate friends, hostiles and unknowns can permit either quicker/easier assimilation of the same quantity of data, or equal quality of discrimination of target types on a greater number of targets, in the same amount of time. The use of color will provide a significant operational benefit to a high capacity data system like JTIDS/TADIL J.

FUTURE IMPLICATIONS

JTIDS/TADIL J is in its infancy as a communications system. Thus far we have reviewed its capability to pass target data between participants on the JTIDS circuits. We have investigated its use with systems and weapons that are already incorporated in the F/A-18A aircraft. However new systems and missions are evolving which can utilize JTIDS/TADIL J data.

- o AMRAAM - Advanced Medium Range Air-to-Air Missile - This missile will give the Navy a new dimension in air-to-air capability. The present AIM-7 Sparrow missile requires that the target and missile be tracked by the host aircraft radar for the entire time of flight. Therefore, the pilot can only shoot one target at a time. AMRAAM gives the capability to consider two or more tracks as possible targets. This will necessitate additional cues to the pilot and additional control messages from a controlling aircraft.

- o GPS - Global Positioning System - With its highly accurate geographical position data, GPS gives the JTIDS community an anchor point for its accurate relative position data. With only a few GPS/JTIDS equipped platforms, many JTIDS platforms will have excellent geodetic position information. This can be of significant benefit in coordinating with ground personnel.

- o Reconnaissance - The navigational benefits provided by JTIDS will enhance performance of the reconnaissance mission. However, as TADIL J is developed, the control capability also will be of great importance. Eventually, the field commander will be able to have a reconnaissance aircraft "on station," and provide commands advising the area to be observed, the type of sensor to use, when to turn it on and off, whether to overfly the area or perform the mission in a "standoff" mode. Obviously, this opens another new dimension for transferring the targeting information/reconnaissance mission results back to the tactical commander for his strike action.

- o Harpoon - The use of an attack aircraft to perform targeting via JTIDS for remotely launched air-to-surface and surface-to-surface missiles, or its own missiles, can give a new capability in war-at-sea missions. Methods for target designation and coordination with remote launch platforms will need to be developed.

- o Onboard Fault Isolation - As aircraft systems incorporate reliable fault detection and isolation capabilities, sending this data to the carrier or airbase during the return phase of a mission could significantly reduce turn-around time between missions. The ground crew could have replacement equipment available for installation in the aircraft while it is being refueled and rearmed, or provide for other necessary maintenance planning prior to return from a particular sortie.

- o ATR - Automatic Target Recognition - As faster computers are developed, the ability to merge data from radars, electro-optical scanners, and infrared detectors to find and identify targets automatically will become a reality. This will greatly increase the amount of target data that fighter/attack/reconnaissance aircraft can transmit.

These few examples only scratch the surface of the potential for great strides that can be made in the field of command, control, and communications thru the use of JTIDS/TADIL J. The war of the future will be an integrated war. JTIDS/TADIL J provides the necessary interface between air, ground, surface, and command units.

CONCLUSIONS

Full implementation of TADIL J to the maximum of its capability in any platform requires a considerable amount of data processing capacity. Much of this is required to filter, prioritize, and store the incoming data. The rest is required to merge JTIDS and onboard sensor data and put this information into formats most meaningful to the pilot.

If too much data is presented to the pilot, without a method of getting down to the "nitty-gritty," they will only use JTIDS for its secure voice channels. To preclude this, the displays

must give the pilot a dimension he has not seen before. This will require extensive simulation and pilot evaluation of the cockpit controls and displays to develop and merge JTIDS information with onboard sensor data in a usable productive fashion. This also will require a committment on the part of the designers and sponsors of all platforms to utilize JTIDS to its fullest - to supply data for other's use and to integrate the incoming data.

Direction needs to be provided in the development of circuits that individual JTIDS platforms use. This is of prime importance for it is ultimately the circuits that a platform participates on that determines the interchange of data between platforms. Providing adequate circuits will permit a level of target information exchange between flight members never before realized. No longer will the pilot have to tell his wingman over the garble of the radio that there is a target at "12 o'clock high," or "you take the one on the left." The data would long since have been displayed in a wingman's cockpit with unambiguous markings identifying his target.

Implementing JTIDS and TADIL J into the modern military communications system is a very complex task: but the reward potential is high.

1. C. H. Hildreth and Bernard C. Nalty, "1001 Questions Answered About Aviation History", New York: Dodd, Mead and Company, 1969, p. 76.

2. Wamble, Thomas L., "Implications of Joint Tactical Information Distribution System (JTIDS) on Fighter and Attack Aircraft Operations", AIAA/IEEE 6th Digital Avionics Systems Conference, Baltimore, 3-6 December 1984.

SESSION 20 MERGED DIGITAL MAP TECHNIQUES

Chairmen:

J.W. Weber
Hughes Aircraft Co.

20

This session addresses the design of merged-data digital map systems, including data compression, and the application of such systems in navigation, threat avoidance, and low altitude penetration.

Digital Map Data Compression Techniques

Paul L. Poehler
DBA Systems, Inc.
11781 Lee Jackson Memorial Hwy.
Fairfax, Virginia 22033

Abstract

Several heretofore uncombined and novel approaches to the multidimensional data compression problem are described, including adaptive lattice finite impulse response (FIR) and infinite impulse response (IIR) filtering, reflection coefficient calculation using a modified Burg harmonic-mean method, adaptive Max quantization, and entropy coding. To demonstrate this technique, a digital 2-D (imagery) compression system was realized and its performance evaluated within the strict design constraints of realizability and near real-time (NRT) operation. This technique is also ideally suited for Mapping, Charting, and Geodesy (MC&G) applications in compression of digital terrain elevation data.

Parameterized numerical results were obtained experimentally for various filter sizes and number of quantization levels. Specifically, when applied to the 2-D problem, results indicate that the 2-D lattice adequately characterize imagery. A new in-place computational algorithm for performing the required FIR and IIR filtering is introduced and demonstrated in 2-D suitable for multidimensional parallel processing. This algorithm provides a significant reduction in the required computation time.

An 80-90 percent reduction in digital data storage requirements is demonstrated when using at least a 4-stage 2-D LPC lattice filter. This performance compares favorably with existing transform-based techniques and requires significantly fewer computations. Finally, these techniques are extended to additional dimensions analytically and are suitable for solution of the general multivariate modeling problem.

I. Introduction

There is currently a high demand for efficient storage methods for large volumes of digital data.[1] Efficient compression algorithms can be devised for imagery using a direct extension of the linear predictive coding (LPC) techniques utilized in speech processing. When used appropriately, these methods achieve significant reductions in multidimensional digital data storage requirements. Standard LPC methodology is extended using modified Burg computational algorithm as described by Marzetta[2] for 2-D data. With appropriate selection of support, these techniques can be extended to realize an efficient data compression system.[3] LPC techniques have been predominantly applied to the general autoregressive modeling problem.[4] These LPC schemes obviate the need for the additional computational overhead required by transform-based schemes.[5]

The multidimensional digital data compression technique devised herein is suitable for direct implementation and subsequent use in pragmatic applications. Noteworthy in this regard is the application of compression of digital terrain elevation data, digitally encoded cultural features, and mission related information in the Defense Mapping Agency format for MC&G data storage/retrieval and data base definition. The multi-dimensional nature of this scheme allows multiple statistically correlated, spatial parameters to be encoded since the technique dynamically utilizes multidimensional parameter correlation properties to achieve data compression.

Experimental technique feasibility was proven utilizing imagery data for convenience. The realization implemented imposed the pragmatic a priori design constraints of near real-time (NRT) operation, realizability using off-the-shelf technology, and the requirement of near-optimal performance.

The resulting system design is depicted in Figure 1. The forward image compression process utilizes a 2-D FIR lattice compression filter matched to the preselected subimage blocks on an adaptive basis. The resultant prediction error filter (PEF) residuals are then quantized and entropy coded for storage. A reverse image compression process is utilized to recover the imagery. There are several important advantages to this technique, in addition to those applicable to the data storage problem. Namely, the compressed data is also suitable for transmission over reduced bandwidth channel, and has the data properties of a pseudorandom white noise process.

This paper will concentrate primarily on the topics of suitability of lattice filtering for multidimensional applications, real - time implementation, and experimentally demonstrates achievable data compression efficiency using actual test imagery data.

II. Multidimensional Lattice Filtering

Lattice filters can be used to solve[6] least squares prediction problems associated with covariance sequence approximations to multidimensional non-stationary problems. Lattice filters are actually a derivative of the standard direct form digital filter representation. The lattice form has gained favor because of several important implementation-oriented attributes such as insensitivity of the filter performance to parameter perturbations, realizability, and the emergence of computationally efficient algorithms for implementation.

The advantages of multidimensional filter representations using the lattice form are:

1) The stability of the FIR and inverse IIR filters are assured if reflection coefficients are appropriately calculated.
2) They are logically extendable to additional dimensions.
3) They are computationally efficient, requiring only a single multiplication and addition per stage.
4) Filters can be designed in the reflection coefficient domain, utilizing fewer realization parameters than the equivalent direct form.
5) The necessary reflection (filter) coefficients can be directly estimated from raw data using a modified form of the Burg harmonic mean algorithm.
6) They can easily be made adaptive since the Gram-Schmidt orthogonalization is performed independently on a selective stage-specific basis.
7) They can be realized via in-place computational algorithms using parallel processors.

In fact, lattice realizations can be viewed as performing a selective orthogonalization transform to the dimension required to maintain PEF residuals below a predetermined tolerable threshold.

Lattice filters are completely characterized by a set of partial correlation (reflection) coefficients. This set of coefficients is smaller than the set of filter coefficients utilized in the direct form realization. A direct relationship exists between these two representations, however (see Figure 2). A lattice filter designed in the reflection coefficient domain has a correspondence to the direct realization with the appropriate associated filter coefficients. The filter coefficients $a(n,m)$ can be obtained directly via a linear recursive combinatorial algorithm from the reflection coefficients $\rho(n,m)$.

A. 2-D Lattice Formulation

The 2-D lattice formulation, as implemented, is a logical extension of the 1-D Burg algorithm and analogously derives a set of 2-D reflection coefficients $\{\rho(n,m)\}$. These reflection coefficients are estimated directly from the input data samples. The 2-D PEF design or realization problem is one of representing the FIR minimum-phase whitening filter $H_{N,M}(z_1,z_2)$ in terms of a finite number of reflection coefficients $\{\rho(n,m)\}$ over an NXM region. The standard autocorrelation (spectral factorization) method was not employed to find the reflection coefficients, since this method is computationally inefficient. The procedure of first forming the autocorrelation estimate was therefore avoided. Each reflection coefficient was chosen sequentially so as to obtain, at each successive stage, the best MMSE 2-D fit to estimate the optimum reflection coefficients.

For example, at the beginning of the $(n,m)^{th}$ stage of the algorithm, the system function is $H_{n,m-1}(z_1,z_2)$. To determine the next stage $H_{n,m}(z_1,z_2)$, the new reflection coefficient $\rho(n,m)$ is chosen to minimize the sum of the new forward and backward prediction errors:

$$\sum_{(k,l)} \sum \{[\varepsilon^{(+)}(n,m:k,l)]^2 + [\varepsilon^{(-)}(n,m:k-n,l-m)]^2\} \quad (1)$$

where $\varepsilon^{(+)}(n,m:K,l)$ is the forward prediction error for the stage (n,m) at the data points (k,l). The forward and backward prediction errors can each be individually expressed in terms of the unit sample response at each lattice filter stage (n,m) relative to $x(s,t)$ as:

$$\varepsilon^{(+)}(n,m:k,l) = [x(k,l) - \sum_{(s,t)} \sum h(n,m:s,t)\, x(k-s,l-t)] \quad (2)$$

and,

$$\varepsilon^{(-)}(n,m:k,l)=[x(k,l) - \sum_{(s,t)} \sum h(n,m:s,t)\, x(k+s,l+t)] \quad (3)$$

In practice, the maximum possible extent of the sample imagery block should be used to gather the data points (k,l). This is essential since the reflection coefficient calculation characterizes the basic statistics of the imagery block of interest in terms of partial correlation of each pixel with its neighbors in an ordered fashion. The PEF filter mask may be run off the edge of the data without adversely affecting the reflection coefficient estimate when adequate filter support is provided. This is particularly true if the dimensions of the input imagery data block significantly exceed the input 2-D PEF mask or equivalently, the lattice filter dimensions. This same consideration is important after the reflection coefficient estimate has been determined and the imagery data is filtered, yielding the PEF residuals (innovations process). In the limit, the optimum MMSE residual sequence will be obtained when the input imagery data block is of infinite extent.

Successive stage forward and backward prediction errors are then given in terms of the estimated reflection coefficients by:

$$\varepsilon^{(+)}(n,m:k,l)=[\varepsilon^{(+)}(n,m-1:k,l) - \rho(n,m)\, \varepsilon^{(-)}(n,m-1:k-n,l-m)] \quad (4)$$

$$\varepsilon^{(-)}(n,m:k,l)=[\varepsilon^{(-)}(n,m-1:k,l) - \rho(n,m)\, \varepsilon^{(+)}(n,m-1:k+n,l+m)] \quad (5)$$

These equations basically constitute the 2-D prediction error lattice filter forward and backward error propagation at each successive filter stage. The $\rho(n,m)$ utilized herein is specifically chosen to minimize the expression for the sum of the squares of the forward and backward prediction errors given by:

$$\sum_{(k,l)} \sum \{[1+\rho^2(n,m)][[\varepsilon^{(+)}(n,m-1:k,l)]^2 + [\varepsilon^{(-)}(n,m-1:k-n,l-m)]^2 - 4\rho(n,m)[\varepsilon^{(+)}(n,m-1:,k,l)\, \varepsilon^{(-)}(n,m-1:k-n,l-m)]\} \quad (6)$$

Taking the derivative with respect to $\rho(n,m)$ to find the MMSE, in terms of both the forward and backward prediction error, and setting it equal to zero yields:

$$\rho(n,m)= \frac{2 \sum_{(k,l)} \sum \{[\varepsilon^{(+)}(n,m-1:k,l)][\varepsilon^{(-)}(n,m-1:k-n,l-m)]\}}{\sum_{(k,l)} \sum \{[\varepsilon^{(+)}(n,m-1:k,l)]^2 + [\varepsilon^{(-)}(n,m-1:k-n,l-m)]^2\}} \quad (7)$$

This is the single basic expression for the desired reflection coefficient estimates from the 2-D sample data at each stage of the recursion in terms of the forward and backward prediction error from the previous stage. In combination with (4) and (5), successive reflection coefficients can be estimated recursively, utilizing the forward and backward prediction errors calculated from the previous lattice filter stage.

B. Reflection Coefficient Calculation Algorithm

This computational algorithm gives an optimal MMSE estimate for stationary processes when the extent of the image is much greater than the extent of the filter $H_{N,M}(z_1,z_2)$. This constitutes the complete 2-D algorithm for the calculation of all of the filter parameters. Addressing the subimage boundary problem at each successive stage of the recursion process yields the following procedure:

Step 1 - Initialization

Let $H_{0,0}(z_1,z_2)=1=H_{0,0}(1/z_1,1/z_2)$

$$P_{0,0} = \text{const} = \sum_{(k,l)} \sum x^2(k,l)$$

$$\varepsilon^{(+)}(0,0:k,l) = x(k,l)$$

$$\varepsilon^{(-)}(0,0:k,l) = x(k,l) \quad .$$

Step 2 - For calculation of the filter stages along columns n=0, $1 \le m \le M$ or $1 \le n \le N$, $-M \le m \le M$, we already have given from the previous filter stage:

$$H_{n,m-1}(z_1,z_2),\ H_{n,m-1}(1/a_1,1/z_2)$$

$$P_{n,m-1},\ \varepsilon^{(+)}(n,m-1:k,l) \text{ and } \varepsilon^{(-)}(n,m-1:k,l) .$$

We first compute the reflection coefficient estimate (7) and then use the update equations for each of the PEF parameters:

$$H_{n,m}(z_1,z_2)=H_{n,m-1}(z_1,z_2)-\rho(n,m)z_1^{-n}z_2^{-m}H_{n,m-1}(1/z_1,1/z_2)$$

$$P_{n,m} = P_{n,m-1}\left[1-\rho^2(n,m)\right]$$

$$\varepsilon^{(+)}(n,m:k,l)=\varepsilon^{(+)}(n,m-1:k,l) - \rho(n,m)\,\varepsilon^{(-)}(n,m-1:k-n,l-m)$$

$$\varepsilon^{(-)}(n,m:k,l)=\varepsilon^{(-)}(n,m-1:k,l) - \rho(n,m)\,\varepsilon^{(+)}(n,m-1:k+n,l+m) .$$

Step 3 - for the recursion between columns for $(1 \le n \le N)$ we have at the image block boundary for conditions:

$$H_{n,-(M+1)}(z_1,z_2) = H_{n-1,M}(z_1,z_2)$$

$$P_{n,-(M+1)} = P_{n-1,M}$$

$$\varepsilon^{(+)}(n,-(M+1):,k,l) = \varepsilon^{(+)}(n-1,M:k,l)$$

$$\varepsilon^{(-)}(n,-(M+1):,k,l) = \varepsilon^{(-)}(n-1,M:k,l) \quad .$$

In practice, this algorithm can be implemented so that the prediction error $P_{n,m}$ is monitored at each filter stage and the algorithm terminated once this error has reached a tolerable level.

III. Real-Time Implementation

In addition to the computational efficiency realized in a digital filter implementation using the lattice filter form, several other multi-dimensional compression techniques were employed. These real-time implementation techniques included:

1) Adaptively performing data compression on a subimage block basis using, if possible, imagery segments of common statistics for best performance.

2) A design which minimizes computational storage requirements - for a 16x16 pixel subimage block requires only 1KB of storage/lattice filter stage. Imagery is decimated into subimages suitable for both real-time and adaptive processing.

3) In-place computational algorithms can be implemented to permit operation on all subimage blocks independently and simultaneously.

A. 4-Stage 2-D Filter Realization

Using the appropriate image support, the 4-stage FIR and IIR filters can be implemented. Statistical trends from this implementation are depicted in Figure 3, which illustrates the behavior of normalized subimage block prediction error vs. the number of lattice filter stages, namely:

1) The normalized prediction error in most cases is reduced by over 90 percent after the first stage, and in excess of 95 percent by the 4th stage for virtually all subimage blocks.

2) The mean and variance of this prediction error on a 16x16 pixel subimage basis is virtually unaffected by 'clip and stretch' dynamic range enhancement and is insensitive to imagery type (for either remotely-sensed or head and shoulders imagery).

3) A 12 stage filter provided some improvement in the residual prediction error beyond the fourth stage.

4) The few isolated large prediction errors occurred on subimage blocks constituting picture backgounds with random pixel variations.

5) Almost all interpixel correlation is evidenced in the pixels immediately surrounding the pixel being compressed.

B. In-Place Computational Algorithm

The 2-D digital lattice filter implementation utilized in-place computations to calculate individual lattice filter stages comprising the 4-stage FIR/IIR filters. In actuality, the in-place software simulation requires slightly more storage than the sequential raster scan filter implementation (all 16x16 pixels instead of just a single row or column of 16). It is also more amenable to NRT parallel processing realizations, which more than compensates for the additional storage requirement.

The manner in which 2-D lattice filters are depicted is reminiscent of the way in which 1-D filters are constructed by Levinson's recursion. The only way in which the 2-D flow graphs differ from 1-D flow graphs is in the presence of delays of the form $z_1^p z_2^q$. These delay elements are implementable if they are causal. They are causal if the image plane points are ordered appropriately. With delay elements of the form $z_1^p z_2^q$, the lattice filter stages are causal if p is less than zero or if p is zero and q is less than or equal to zero (this designates points in the 'past'). Thus, the flow graphs depicted for the (N,M) stage filters are implementable, and, in fact were implemented for the 4 and 12-stage cases.

In practice, a direct implementation of the lattice filter requires a minimum image data buffer size of one whole row or column of pixels. This is specified by a delay of the form z_1^{-I}. The size of the row or column is the same as the size of the column of image data in the input data array to be computed. (For our case, one 16 pixel buffer is adequate, since subimage blocks are 16x16 pixels in size.)

For the alternate in-place realization utilized herein, a complete subimage block of 16x16 or 256 pixels is stored and permits simultaneous computation of the subimage residuals at any stage. Selective analysis of prediction error residuals for each subimage block can therefore be performed on a filter stage-specific basis. In fact, 'subimages' containing only the selected stage residuals can be printed and utilized for image region classification purposes. This can be utilized to group subimages of similar statistics and improve the compression ratio by eliminating additional reflection coefficient storage overhead for subimages within a common region.

The numerical computations for FIR lattice filter are performed at the first stage of the lattice filter using the original imagery data x(k,1) as input. The input is fed in a serial raster scan fashion to the filter. Each successive stage performs filtering according to the lattice filter flow diagram, resulting in forward prediction error at pixel x(k,1) of:

$$x_{fe}(k,1) = x(k,1)+k(n,m)x(k-1,1) \qquad (8)$$

and a backward prediction error of:

$$x_{be}(k,1) = k(n,m)x(k,1)+x(k-1,1) \qquad (9)$$

respectively, at stage (n,m) where the partial correlation coefficient $k(n,m) = -\rho(n,m)$. These numerical calculations were performed in an iterative fashion on a stage-by-stage basis using the in-place algorithm for each subimage block.

Numerical data required for FIR 2-D filter support is limited to a border surrounding, on the average, two sides of each subimage block for a total of 2M pixels. For the 4-stage lattice, a one pixel border is required. This border serves as an initial set of boundary conditions for inverse (IIR) filtering. Therefore, it must be stored along with the compressed PEF residual data on disk or a null border can be used.

In actual practice, due to the finite extend of the 16x16 data block, the support border was changed after the zeroth stage of iteration from the actual raw data values to the first order residuals for each successive lattice stage. This more closely approximates the data block statistics, since over 90 percent of the residual prediction error is removed in the first lattice filter stage. This support is treated in an inverse fashion during the 2-D IIR filtering process to recover the original imagery.

Likewise, the inverse 2-D filter was implemented using in-place computations. Assuming appropriate support and negligible round-off error, the inverse filter yields the original image exactly from PEF residuals. In actual data compression applications, however, quantization effects create irreversible distortion which can be minimized, but not eliminated. This is done by appropriate selection of quantization levels to be optimum in the MMSE sense using the Max[7] quantization algorithm.

The inverse filter requires a border of support data of minimum width M around each subimage block. The only remaining adroit manipulation required to affect realization is in the mosaicking of the 256 subimage blocks to form the final reexpanded composite image. This meets the finite image extent boundary conditions while simultaneously furnishing the required subimage block support for inverse LPC filtering.

The numerical computations for IIR lattice filtering are performed at the first stage of the inverse lattice filter using the original imagery data x(k,1) as input. The compressed imagery data (PEF residuals) are fed into the filter in a linear raster scan fashion. Each successive filter stage performs filtering according to the lattice filter of Figure 2, resulting in a forward prediction error of:

$$x_{fe}(k,1) = x(k,1) - k(n,m)\ x(k-1,1) \qquad (10)$$

and backward prediction error of:

$$x_{be}(k,1) = x(k-n,1-m) + k(n,m)\ x(k,1) \qquad (11)$$

respectively, at stage (n,m) where the partial correlation coefficient $k(n,m) = -\rho(n,m)$. These numerical calculations were performed in an iterative fashion on a stage-by-stage basis for each image block. Note that consistent with Morgan's theorem, and in comparison with the forward (FIR) 2-D lattice filtering, the in-place computational technique requires the complete propagation of the forward prediction error from stage (N,M) to (0,0) followed by the propagation of the related backward prediction error from stage (0,0) back to stage (N,M) to regenerate each successive pixel value. Each new pixel value then also generates the required corresponding forward and backward prediction error residuals, utilized successively to generate the next pixel value. This inverse in-place IIR filtering process continues until the complete subimage block has been regenerated. On retrieval, the support required for the 2-D inverse IIR lattice filter is obtained directly from the disk along with the compressed PEF residual data for each subimage block. Without this support the image data cannot be recovered exactly from the PEF residuals alone.

Quantization of PEF residuals was accomplished according to the Max algorithm. Max quantization input and output levels are tabulated for standard Gaussian and Laplacian input data distributions. For this adaptive scheme, however, optimal Max quantization thresholds were calculated using actual PEF residual statistics. The midriser quantizer has an even number of output levels and was employed since an even number of levels results from the 1, 2, 3, or 4-bit quantization simulated. No specific a priori block statistical characterizations were made. This adaptive quantization scheme determined the PEF residuals on a block-specific basis for optimal MMSE threshold placement.

Entropy coding was accomplished after PEF data quantization. The encoding tree was determined from the frequency histogram of the actual quantized imagery data values in each subimage block. The companion inverse decoding operation is accomplished via this tree upon data retrieval. The decoding tree must also be stored on the disk on a block-specific basis. Entropy coding affords significant additional data compression margin even including the minimal data storage overhead required to store the encoding tree for image data recovery purposes.

Entropy coding, therefore, gives a dramatic improvement in the achievable data compression factor in the final stage of the 2-D LPC imagery compression process. Unlike Max quantization,entropy coding does not introduce any distortion and is completely reversible. In fact, with the cascade use of this method, after lattice filtering and quantization, image compression is possible to an average number of bits/pixel commensurate with the best spatial transform compression schemes.

IV. Data Compression Efficiency

The resultant data compression efficiency achieved through utilization of LPC techniques is governed primarily by the number of quantization levels employed and whether entropy or equivalent coding is utilized. For 8-bit pixel data, if four gray levels are permitted, the compression ratio without entropy coding is 4 to 1 using 2-bit compressed pixels and can be further reduced to under 2 bits/pixel with entropy coding. There are minimal overhead storage requirements imposed by the LPC algorithm for data reconstruction purposes which bear mention and must be considered on implementation. This overhead data is required on a block-specific basis for imagery reexpansion, and is delineated as follows:

1) 2-D Lattice Filter Support Data - A border of data in the 'past' of the ordered filter direction is required to recover the original imagery data from the PEF residuals. This border is not large, and its size is governed by the dimensions of the 2-D filter mask employed. For the 4-stage FIR/IIR filter used herein, a 16x2 pixel border was required on the average for each subimage block. This amounts to a small percentage of the actual number of pixels filtered (10 percent) and is a smaller percentage for larger size subimage blocks.

2) Reflection Coefficients - For an N-stage 2-D LPC filter there are N reflection coefficients per subimage block required to realize the FIR and IIR compression filters. The 4-stage therefore has a subimage block overhead of 2 percent.

3) Quantization Level Data - This data is merely an ordered list of the numerical values of the Max quantization thresholds used to quantize the data. This permits an ordered and accurate reexpansion of the imagery data by block. Again, the overhead margin herein is relatively insignificant. For the 4-level quantizer, 4 numerical values are required for each 256 pixel image block for a margin of approximately 2 percent.

4) Entropy Coding Data - An entropy tree decoding table is required to permit recovery of the entropy coded data. This overhead is insignificant for the 4-level quantizer and its precise value is dependent on imagery statistics.

The compression factors achievable using adaptive LPC techniques are naturally addressed on a block-specific basis. Obviously, the absolute data compression factor realized over the entire image is dictated by both the imagery statistics and the size of the image subblocks chosen. The block size should be determined by the second-order statistical behavior of the image hence the adaptive nature of this block quantization scheme. Regions of common behavior can be combined and afford attendant savings of the aforementioned block-specific overhead, resulting in a better overall compression factor.

The block size utilized in these investigations was purposely chosen to be small in order to demonstrate the variations of statistical behavior within an image and fundamentals of the adaptive LPC data compression technique, rather than to achieve the best compression factor. A system which uses larger image blocks and/or can recognize and combine smaller blocks of common statistics, will realize a much higher data compression ratio in practice.

Efficiency Using Entropy Coding

Data compression efficiency was estimated statistically using first and second order Markov process entropy statistics. These statistics were empirically calculated on a selected subimage block basis. The N^{th} order entropy is given by:

$$H_N = -1/N \sum_i P(B_i) \log_2 P(B_i) \qquad (12)$$

where $P(B_i)$ is the probability of a sequence B_i of N picture elements occurring. The N^{th} order entropy is a monotonically decreasing function which converges to the true image entropy given by:

$$\lim_{N \to \infty} H_N = H \qquad (13)$$

Using these equations, the average number of bits/pixel (average sequence length) along with the theoretical entropy limits (lower bound on compression efficiency) were calculated for both low and high activity subimage blocks and graphed in Figure 4. Imagery Max quantized to 3 or 4 bits/pixel and reduced by entropy coding to 1-2 bits/pixel hase an excellent rating in terms of resultant compressed image fidelity, which is commensurate with the best transform compression techniques. Additional potential for increased data compression factors exists with the addition of entropy coding (on the order of 50-60 percent).In fact, 2-D LPC lattice filtering reduces the storage requirement alone by 50-60 percent, and subsequent entropy coding by an additional 50-60 percent. Since the 2-D LPC lattice filtering effectively reduces the image dynamic range, any post-LPC compression technique such as DPCM encoding on an image line or subimage block basis is attractive. This data clearly demonstrated that the range of pixel compression efficiency was from 50-60 percent reducing the 4 bit quantized image to an average of 2.5 bits/pixel to over 90 percent at .6 bits/pixel.

A. Performance Tradeoffs

Several conclusions are evident from the performance tradeoff criteria between 2-D LPC performance and number of Max quantizer levels.

1) The compressed image is very sensitive to subimage block selection. In particular, abrupt intensity changes should form subimage block boundaries for reflection coefficient calculation and subsequent LPC filtering. The compression process is therefore sensitive to edges.

2) The number of Max quantization levels is not as important as the number of LPC filter stages, until at least 3 or 4 stages are employed. After that point, little prediction error improvement is obtained, and increased performance in terms of image fidelity is primarily governed by the number of quantization levels employed.

3) Subimage blocks were chosen here without regard for image content specifically to investigate 2-D LPC lattice compression performance characteristics. Better image fidelity can be achieved if prior image statistical segmentation is performed to determine the specific subimage regions to be compressed.

B. Improvement Methods

The following methods of improving 2-D LPC imagery data compression performance are therefore evident:

1) A priori image region classification via reflection coefficient statistics and/or Laplace filter edge detection. Larger regions can therefore be characterized by fewer reflection coefficients, reducing the data storage overhead, and eliminating the leakage and blocking effects evident in high-contrast regions.

2) Accommodate an adaptive number of 2-D LPC stages per subimage block to maintain prediction error residual below a predetermined image fidelity-based threshold. (Upon expansion, the corresponding number of inverse 2-D IIR LPC lattice filter stages must also be used.)

3) Use of an image enhancement technique to remove 2-D LPC lattice filter-induced artifacts. A typical simple technique is low-pass Gaussian filtering. This technique can maintain a given image fidelity to a 10-20 percent greater compression factor.

4) Use entropy coding to achieve a 50-60 percent reduction in the number of bits required to store the imagery.

5) Perform a log compression on the imagery data to reduce its dynamic range prior to 2-D LPC lattice filtering.

6) Use a deblocking enhancement filter to reduce and/or eliminate residual subimage block compression artifacts.

V. Conclusions

Conclusions which can be drawn to summarize these 2-D results as they apply to modeling the general multivariate problem are:

1) A realizable, efficient, and readily implementable 2-D realization provides performance comparable to existing transform-based techniques such as the DCT. For example, an 80-90 percent data compression factor was realized using 1-4 stages of lattice LPC filter, with performance within .5 bits/pixel of the theoretical bound.

2) This 2-D realization confirms that reflection coefficients provide a good means of characterizing imagery.

3) A new in-place computationally-efficient parallel processing technique was proven in 2-D and is applicable to the multidimensional problem.

4) The forward (FIR) and inverse (IIR) 2-D lattice filter support existance conditions have been verified experimentally and, with careful ordering, applied to multiple dimensions.

A. Implementation Considerations

The primary new conclusion, with respect to hardware implementation, was to employ parallel processing techniques for in-place FIR and IIR computations on a subimage basis. Each subimageblock can also be processed for reflection coefficients and PEF residuals simultaneously. The forward and backward correlation process performs essentially the required lattice filtering.

In addition, minimal additional requirements exist for imagery data buffer storage. Buffer sizes commensurate with the selected subimage dimensions for both forward and backward error prediction per stage are required. For example, for the in-place 2-D 4-stage lattice computations, 2 KB of data buffer storage was required.

An additional implementation-oriented concern is the adaptive variation of subimage block size based on imagery statistics. This would permit more efficient operation, but requires a priori or a posteriori decision hardware to estimate imagery statistics prior to PEF residual storage on disk. The most efficient and least-risk path is to calculate reflection coefficients a priori on a subimage basis and then recognize block combinations utilizing the root-mean-square (rms) combined values of the respective subimage reflection coefficients. The hardware augmentation to implement this scheme would be minimal and be more than compensated in improved performance with most imagery.

B. Performance Prediction

The performance of any 2-D LPC compression system is highly dependent upon the following prediction parameters:

1) Imagery Dynamic Range - Expressed in the number of bits per pixel. The 2-D LPC algorithm will produce improved compression ratios with imagery in excess of 8 bits per pixel. This is commonly found in Landsat multispectral scanner and synthetic aperture radar data, which can range from 10 to 18 bits per pixel.

2) Size of Subimage Blocks - The larger the size of subimage block, the better the performance in terms of compression factor improvement. As the subimage block size is increased, however, the block second-order statistical estimate in the form of reflection coefficients becomes more inaccurate. There is, therefore, an optimal subimage block size which can be determined dynamically, but at a great additional complexity.

3) Number of Lattice Filter Stages - Little, if any, performance improvement is gained for most imagery beyond the 4-stage lattice. It is also possible to vary the number of 2-D lattice stages dependent upon the calculated prediction error residual so as to guarantee an a priori percent prediction error below a predetermined threshold.

There is no closed-form analytical expression which represents the numerical value for a 2-D LPC performance measure in terms of image fidelity. Rather, each image bears its own statistical variations; hence, the observations above, in combination with the adaptive block quantization scheme realized, is the best performance prediction guide.

C. Multidimensional Lattice Filtering

A significant conclusion reached via this 2-D LPC analysis and implementation is that similar carefully ordered approach can be applied to reflection coefficient calculation and lattice filtering in an arbitrary number of dimensions.