MULTIPLEX SYSTEMS SIMULATOR

J. E. Camp, AFAL, Wright-Patterson AFB, Ohio
J. A. Gracia, Harris Corporation
Electronic Systems Division, Melbourne, Florida

ABSTRACT

Multiplex Systems Simulator, known as MUXSIM, is a software package which performs computer-aided design and design verification of digital information transfer/multiplex systems. It is primarily an aid for an organized approach at specifying and designing multiplex systems for diverse applications. MUXSIM is a useful tool for determining answers to some complex digital multiplex system implementation problems, such as the following:

- Command and control techniques for data transfer
- Data bus requirements
- System interface hardware requirements
- Data processing and manipulation requirements

MUXSIM was designed by Harris Electronic Systems Division for the Air Force Avionics Laboratory (AFAL), Air Force Systems Command, United States Air Force, Wright-Patterson AFB, Ohio. The simulator system is required by AFAL to aid in the design and development of the more sophisticated data transfer/multiplex systems necessitated by the trend towards more totally integrated digital avionics information systems for the aircraft of the 1980's. However, its implementation is general in nature, enabling it to address any of a family of transfer requirements where information originating from many sources spatially separated must be arranged and/or processed, then retransferred to an equally complex destination arrangement.

Essentially, the MUXSIM programs serve to help bridge the gap between detailed analysis and prototype hardware. They provide a means of interplaying the pieces of the detailed analysis (such as update rate requirements, sampling requirements, data buffering requirements, data bus rate requirements, processing and time delay requirements) from a myriad of point-to-point signals into a coherent requirement which can be verified for compatible performance prior to attempting a hardware development program. This approach can prevent costly system errors from being implemented into hardware. MUXSIM was designed to supplement and complement the other tools used in multiplex system design.

INTRODUCTION

With the constant growth in aircraft sophistication and corresponding additional equipment requirements that has been evident in the recent past, the use of point-to-point wiring to carry signals between different aircraft sensors or subsystems (or within the subsystems) has become an increasing problem to the aerospace industry. The weight penalty and cost of installation and modification has created a negative impact on the aircraft's performance and economy.

In recent years, a move toward shared resources techniques - multiplex technology - was started. However, it became apparent after a short involvement with the problem that there were many approaches and methods for implementing the sharing of resources to transfer signals back and forth between the various subsystems. Confusion arose as to the choice among approaches proposed to solve the problem. While some answers were readily evident from current technology, the availability of hardware, and/or ease of implementation; other answers were not so easily derived. AFAL recognized this state of confusion to be a major problem and decided to construct a simulator to help arrive at additional answers to this complex problem. The result of this effort is a software package which performs computer-aided design and design verification, including simulation of digital information transfer systems. The package, primarily written in FORTRAN, uses GASP IV[1] as a simulation language and currently implemented on a DEC System-10.

MUXSIM OVERVIEW

An overview of the MUXSIM System can be derived from its user or simulator operation concept shown in Figure 1 and described by the following steps:

- The MUXSIM user configures the aircraft system he would like to multiplex by using existing data from the workload library or by generating the information and submitting it in bulk form.
- Next, using MUXSIM he selects the model of the specific multiplex system that he would like to use in the aircraft from the System Configuration Model Library.
- All along, the user interacts with the MUXSIM System via TTY/CRT console in regard to such factors as choice of models, choice of model parameters, adjustment of inconsistent data, etc.
- MUXSIM takes over, runs the simulation experiment, and informs the user of the results of his experiment.
- The user reviews the results and selects desired changes, such as choice of models, choice of model parameters, etc.
- The process is iterated until he is satisfied that the multiplex system selected has been optimized for the specific application.
- At the conclusion of this process the user has a large volume of design information which relates directly to the selected system. This data may be stored in a data file for further use and selected portions may be presented in hard copy using one of the available edit options.

An equivalent but somewhat different view of MUXSIM is shown in Figure 2. It provides a more detailed picture of how the MUXSIM's three major elements - inputs, models, and outputs - interplay.

MUXSIM DESCRIPTION

MUXSIM description is continued from three points of view: the functional, the logical, and the operational. These viewpoints are intended to reflect the requirements of the MUXSIM user, the MUXSIM programmer, and the MUXSIM host computer, respectively.

Functional Description

A short, but to the point, description covering the user's point of view was carried forth in the preceding section "MUXSIM Overview." To amplify the functional description of MUXSIM, the following is postulated as an example of the use of MUXSIM during a hardware development program.

- In the conceptual stages, past data bases developed by MUXSIM for similar systems are useful for bounding the requirements.
- As the development program matures, MUXSIM uses the emerging information to help develop the peculiar system's actual data handling requirements.
- As candidate systems emerge, they are modeled and evaluated to indicate specific relative merits and problems of each. A final selection is based on how these advantages and problems affect the application.

Therefore, from a functional or user's point of view, MUXSIM was designed as an extensible software package intended to adapt to the multiplex system designer's requirements. MUXSIM has expanded the user's multiplex systems design verification capability from rudimentary accounting techniques, the normal means available to manual verification, to include (discrete event) simulation for a more in-depth verification.

Logical Description

From a logical or software implementer's point of view, MUXSIM consists of four major subsystems (see Figure 1), termed the Utility, Static, Dynamic, and Executive. These subsystems are divided into a total of 26 interrelated programs which are controlled by the Executive subsystem's main program.

The Utility Subsystem is essentially the data base management system for MUXSIM, where the data base consists of one or more signal flow lists which characterize the information transfer workload to be handled by the multiplex system being simulated.

The Static Subsystem consists of the remote terminal (RT) assignment, word map, message map, fixed-format scheduling, and fixed-format bus loading computation. It contains eight models which represent eight different hardware configurations for word/message mapping. This subsystem is called "static" because dynamic handling of simulated time is not required.

The Dynamic Subsystem presently consists of two discrete-event models and is called "dynamic" because stochastic events characterizing such phenomena as multiplex system component failures, bus noise, and time-variable data transfer requirements are explicitly considered. This system is quite general, since it uses a complete discrete event and continuous simulation package, GASP IV, as a component. Both Static and Dynamic Subsystems are designed using a modular concept which allows new models to be easily added by the Advanced User of MUXSIM.

The Executive Subsystem provides the interface between the user and the other three subsystems; in addition, it provides the interactive coached environment which is the key to MUXSIM usage.

Operational Description

From an operational or computer center manager's point of view, MUXSIM is a fairly large software package. Physically, it consists of about 9,500 FORTRAN statements (this does not include the GASP IV requirements, an additional 2,200 statements). It is normally used in conjunction with large data bases (signal lists) which serve to define the workload. At present, a signal list which comprises 11,300 cards and defines about 2,650 signals used to characterize the information transfer requirements for an A-7D aircraft is being used as a test case. An additional signal list for Integrated Digital Avionics for Medium STOL Transport (IDAMST) is being prepared. Other signal lists exist which comprise about 13,000 to 16,000 signals for aircraft such as B-1, 747, etc.

MUXSIM is presently implemented on the AFAL DEC System-10 under the TOPS-10 operating system. However, because MUXSIM is a FORTRAN-based package, it can be moved from one host system to another if the intended new host system has a FORTRAN compiler and a suitable Disk Operating System. As an example of its portability, the original version of MUXSIM is also currently operational on the Harris Datacraft 6024/5 system, which has a 32K 24-bit word memory.

Costs of "typical" simulation experiments, estimated by conventional means, were found to be $50-$400 for the simulation experiments described herein under "MUXSIM Benefits." These estimates are intended as "ball park" figures only, since the costs will vary with different simulation experiments and with different computer resource accounting methods. While cost-effectiveness of MUXSIM is not rigorously demonstrated by this, it shows that costs of certain simulation experiments felt to be typical are reasonably economical.

USER BACKGROUND

One key element important to the user and not readily obvious from Figure 1 is that MUXSIM is designed to run interactively with optional step-by-step coaching for maximum ease of learning and use by the MUXSIM Basic User. Like any other tool, MUXSIM can be employed to best advantage by those with appropriate background and training. It is assumed that the MUXSIM Basic User has some knowledge of multiplex system design and avionics systems; such a person can realize useful results from MUXSIM with very little training in the operation of MUXSIM itself.

A person interested in more sophisticated exercises, the Advanced User, will require some specific training in MUXSIM and its components. He also requires a working knowledge of FORTRAN to create new static models; to create new dynamic models he must familiarize himself with details of GASP. In short, although MUXSIM is designed to be easy to use, in more sophisticated applications it still requires a good human driver.

Furthermore, although MUXSIM may be modified or transferred to another host system with relative ease, for these operations the user requires some special knowledge of MUXSIM. Two documents, the MUXSIM Users Manual and the MUXSIM System Modification Design Data Manual, contain the information needed for any advanced use or modification of MUXSIM.

THE EVOLUTION OF MUXSIM

The MUXSIM Program has been underway since March 1973. It has, in that time, been influenced by experience from several hardware programs (B-1 EMUX, DAIS, Shuttle PCM, etc.) and has, in turn, influenced some of them.

MUXSIM is being evolved by a four-phase program: Definition (Phase I), Design (Phase II), Construction (Phase III), and Operational Evaluation (Phase IV). The program is at present finalizing construction and making plans for the Operational Evaluation effort which will follow shortly.

MUXSIM Definition. Phase I was accomplished in four subtasks. The first subtask consisted of formalizing the multiplex system design process. (This resulted in an extensive effort categorizing the multiplex system types, the elements of these multiplex systems, methods of their descriptions, and methods for describing their interactions.) The second subtask, conducted in parallel, was a Simulation Technology investigation. This subtask was necessary since maximum use of existing tools must be considered for overall cost-effectiveness. This effort consisted of reviewing various types of simulators, simulation languages, simulator structures, benchmark tests, modeling techniques, and implementation methods for applicability. The third subtask was to identify significant design questions best answered by simulation. This consisted of determining the significant questions, determining when they arise in the design process, and determining the means available for answering them. Finally, the fourth subtask was that of identifying MUXSIM functional requirements, including simulator type, inputs/outputs and variables of simulation, parameter formats, simulator control mechanisms, and real versus simulated time requirements.

MUXSIM Design. Phase II consisted of three subtasks: system functional design, subsystem detail design, and creation of the MUXSIM Functional Design Specification. The design of MUXSIM utilized mainly a top-down approach, with emphasis on modularity. In certain instances, a probe-coding approach was used for sizing and feasibility studies. The probe-coding served as a software breadboard which eventually resulted in improved final software.

MUXSIM Construction. Phase III is being conducted in three subtasks: a software system overall design task, a software module (program) design task, and an integration and verification task. At present, the integration of the MUXSIM System is completed and the system is undergoing an extensive verification.

MUXSIM Operational Evaluation. The IDAMST Signal List effort, now under way, is the start of Phase IV.

Among the critical MUXSIM design decisions which evolved during the Definition and Design phases were the following:

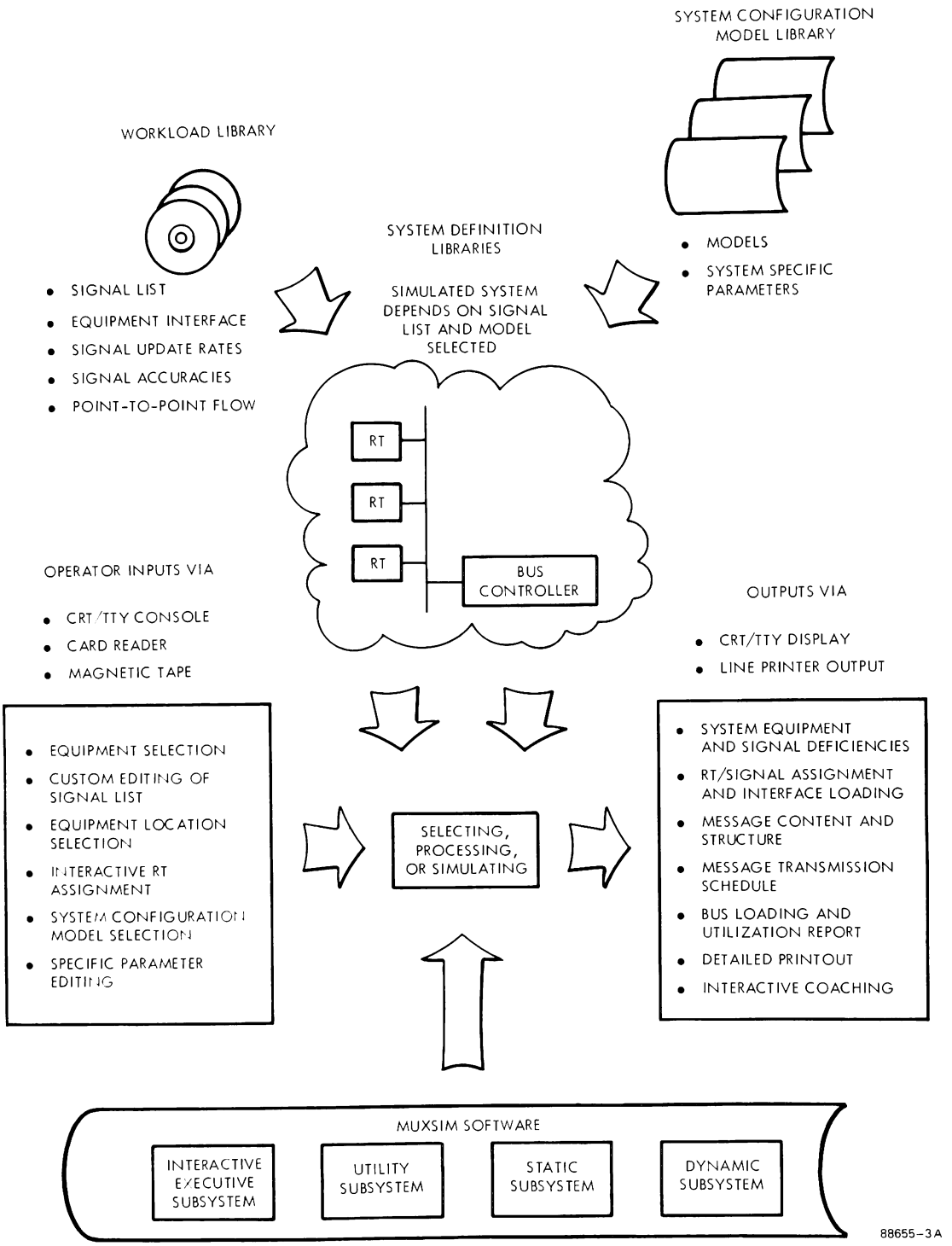


Figure 1. Simulator Operation

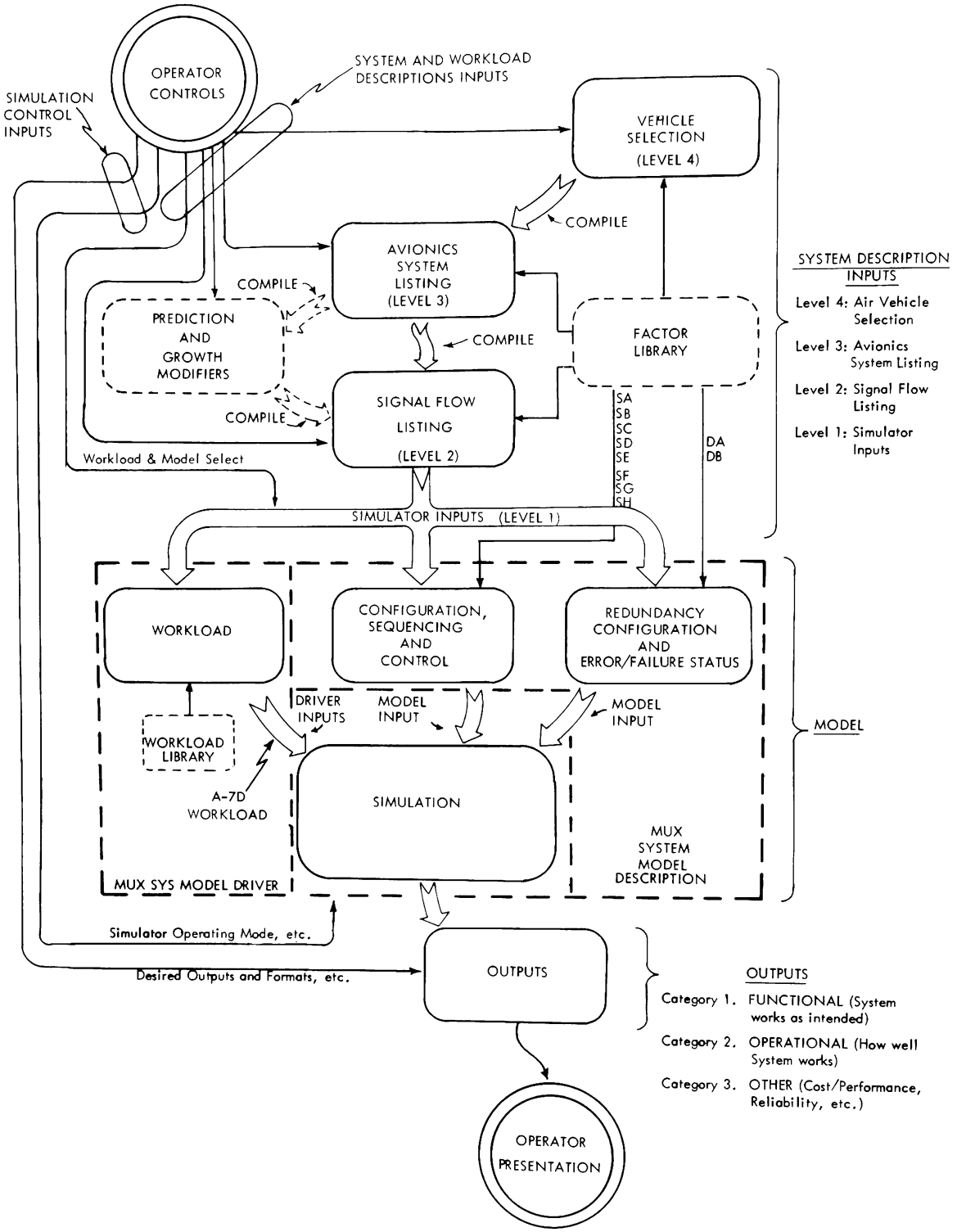


Figure 2. MUXSIM System - Conceptual Level

- Decision to focus MUXSIM on questions asked during the earlier phases of a typical multiplex system design effort (for maximum cost-effectiveness)
- Decision to focus MUXSIM on questions relating to the multiplex system architecture and organizational levels rather than on detail levels such as the electrical design of the bus (for maximum cost-effectiveness)
- Decision to make MUXSIM a software simulator (for maximum response to multiplex hardware changing environment)
- Decision to make MUXSIM interactive (for ease of use)
- Decision to make MUXSIM FORTRAN-based (for ease of development, ease of modification, and "portability")
- Decision to use GASP as a vehicle for implementing dynamic MUXSIM models (for ease of development, and generality)
- Decision to emphasize the Utility subsystem and the development of real workloads for driving MUXSIM models (for credibility of results and realism)
- Decision to make MUXSIM highly modular (for ease of development and flexibility)

MUXSIM BENEFITS

The question addressed here is, "Of what value is MUXSIM?" This is answered by summarizing some general benefits and by describing some specific simulation experiments that have been run using MUXSIM, together with their results.

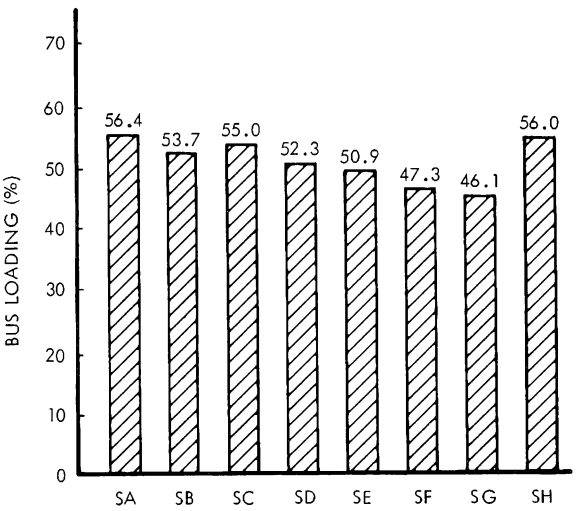
The major MUXSIM benefits can be summarized as follows:

- Computer bookkeeping, accounting, and summarizing of large signal lists which would otherwise have to be accomplished by error-prone manual means. (Note: the test case, the A-7D signal list, consists of about 2,650 signals. Larger aircraft signal lists consist of between 13,000 and 16,000 signals; e.g., 747, B-1.) Computerizing this effort is a logical extension of industry-wide automatic wire lists applications.
- Simulation and/or analysis for in-depth verification of a hardware system's performance under a more exact representation of the actual workload.
- Economical, rapid documentation of key information required to carry out hardware implementation of multiplex systems. In time-division multiplex (TDM) it is necessary to know origin and destination of each signal in order to wire the remote terminals (RT). It is also necessary to know the bit(s) to word, message, and update interval assignment of each signal in order to write the encoding and decoding multiplex system operation software. The documentation desired is selected from standard formats normally used (i.e., signal lists, RT I/O interconnect lists, signal-to-message assignments) and can easily be further edited into the user's particular desire.

Since MUXSIM presently stands somewhere between the construction and the operational evaluation phases, the real benefits have yet to be proven. However, as a part of the MUXSIM System software verification effort, several sample simulation experiments have been performed which illustrate typical results which can be easily obtained by the MUXSIM Basic User. The following experiments, briefly summarized below with their results, are typical of the outputs which MUXSIM will provide.

Experiment 1 - Bus Loading Versus Bus Command and Control Schemes. In this experiment, it was assumed that the multiplex system designer wishes to know which is the best bus command and control scheme for a given bus workload. He also wishes to know how much better one scheme is than another. For example, if a simple scheme is nearly as good as a more complex one, it may be preferable to use the simple one because it may be considerably less expensive to implement.

The present MUXSIM implementation consists of eight static models which represent eight different bus command and control schemes. This implementation covers a wide variety of configurations applicable to TDM systems, ranging from completely centralized (terminal-to-central-to-terminal) to completely distributed (direct terminal-to-terminal), and includes a number of hybrid combinations of both. Figure 3 indicates the results of an experiment run using the A-7D data base, which is being used to verify MUXSIM.



STATIC MUX MODELS

Model Name	Information Transfer Discipline
SA	T/T Transfer
SB	T/C/T Transfer (bit shuffling)
SC	Digital T/T, Discrete T/C/T
SD	Hybrid Transfer
SE	T/T Transfer with BCIU Broadcast
SF	T/C/T Transfer with BCIU Broadcast
SG	Hybrid Transfer with BCIU Broadcast
SH	T/C/T Transfer (word shuffling)

Figure 3. Bus Loading Versus Bus Command and Control Schemes

Experiment 2 - Bus Loading Versus Bus Speed. This experiment assumed that the multiplex system designer wishes to explore details of suspected bus saturation effects. These effects take place in the vicinity of, or close to, 100 percent bus loading. The presence of saturation implies that for practical purposes a certain percent of the apparent bus capacities are unusable for the given bus scheduling algorithm. To investigate these saturation effects, bus speed is systematically varied while the bus workload and command and control schemes are held constant. (This is analogous to expanding the workload.)

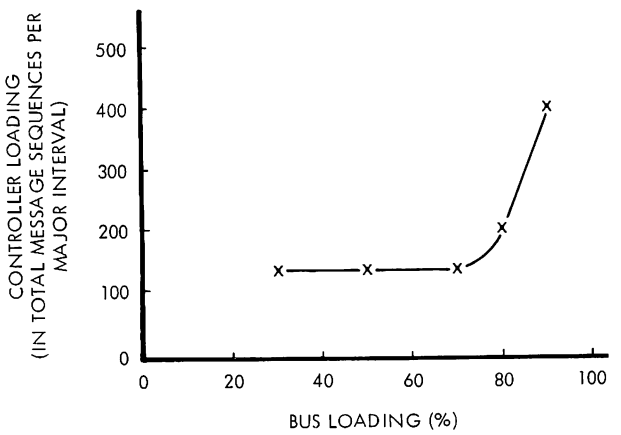


Figure 4. Controller Loading Versus Bus Loading

The final results for the saturation experiment, which used a binary matrix scheduler, show that several fundamental update intervals within the given major frame become saturated at 92.9 percent overall bus loading for the particular implementation tested.

Experiment 3 - Controller Loading Versus Bus Loading. In this experiment it was assumed that the multiplex system designer wishes to explore the impact of high bus loading on controller loading. He suspects that as bus loading increases, the controller may have to work harder in order to implement the given fixed command and control algorithm. As in the preceding experiment, bus loading is increased by decreasing bus speed while maintaining constant workload and bus command and control scheme. The controller loading is measured, for this experiment, by counting the number of different message group sequences that must be handled by the controller as the bus load is varied. The results of this experiment are given in Figure 4.

Experiment 4 - Impact of Command and Control Uncertainties on the Periodicity of the Fundamental Update Interval Starts. This experiment involved the use of a GASP-based dynamic model. The results are illustrated by a GASP histogram plot. For this experiment, a bus load which consisted primarily of the periodic messages plus background demand messages, and a command and control scheme which consisted of addressing a terminal and waiting for a terminal to respond, were assumed. The delays in terminal response could conceivably cause the start of an update interval to be delayed until a response is received from a terminal. The response time variations are attributed to several factors, including clock variation and problem-caused variations such as failures of the response mechanism, whereas the bus controller has to await the timing out of a watchdog timer, which disables the terminal from responding, before the bus controller is free to proceed. This delay can cause the scheduled start time for the next message to be delayed, thereby impacting the start of the next update cycle or fundamental update interval. Should this happen, it delays the start of every message in that interval by that amount.

The histogram in Figure 5 shows statistics of the update interval start time jitter (expressed in fractions of the update interval duration) measured while running this particular experiment.

Experiment 5 - The Impact of Noise on Redundant Buses. This experiment used a dynamic model. It consisted of measuring noise hits on either one of the two buses postulated for this experiment, or on both buses simultaneously. The assumption is that it takes a simultaneous hit on both buses to disrupt the message transmission, thereby causing a loss of the message(s) being transmitted during the simultaneous hit. Table 1 shows the quantity of hits measured during the simulation experiment and the corresponding number of messages lost. In this particular experiment the noise data is not as realistic as the data used in other experiments. However, this is a question of being able to better define the noise data.

Table 1. Results of Noise Impact on Redundant Buses

Number of Messages Sent	=	106,489
Number of Messages Hit on at Least One Bus	=	1,171
Number of Messages Lost	=	100

IMPLEMENTATION OF MUXSIM RESULTS

This subject is typified by the question, "Will MUXSIM's output influence the hardware implementation?" While MUXSIM is at the present time starting into its operational evaluation phase, there are two factors which point to the usability and actual implementation of results derived from the multiplex system simulator.

- MUXSIM was designed as a computer-aided design tool; its outputs are in terms of realistic representations of each individual signal and are mapped into representations of the actual hardware implementation. That is, the outputs are not restricted only to displays of statistical results, but rather they are mechanized to give traceability to the actual signal and its representation.

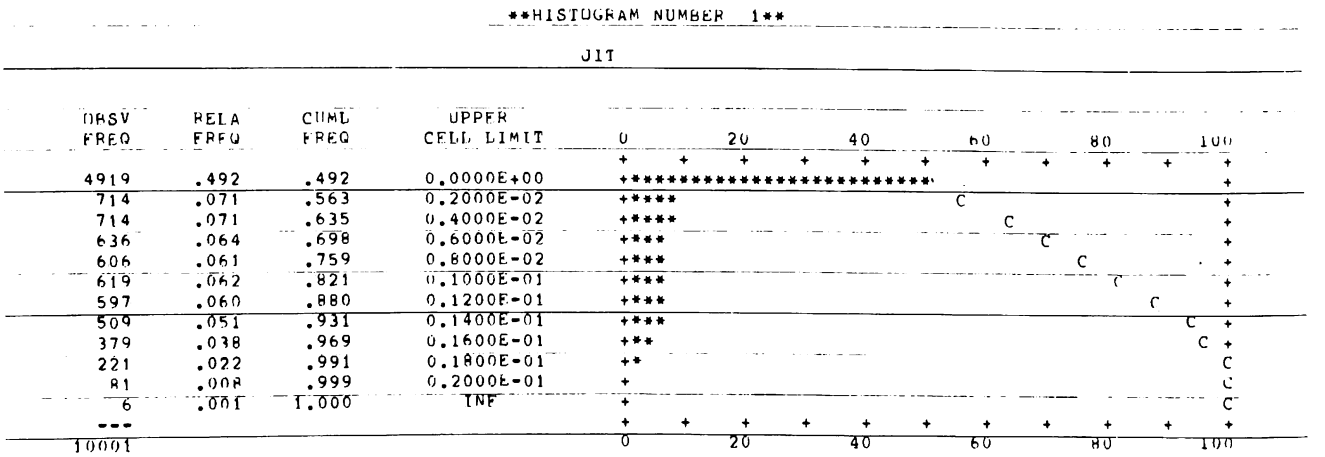


Figure 5. Statistics of Update Interval Start Time Jitter

- Software packages have been used successfully on the B-1 EMUX System and the Shuttle PCM System to derive data bus controller operational software programs. These software packages have also been used successfully to mechanize the implementation of results on their respective hardware development programs. The difference between MUXSIM and these two software packages is that a specific hardware system implementation had already been established and the two programs were instrumental in detailing the actual implementation. MUXSIM is first directed at determining the best system approach; it can then be directed at detailing the actual implementation.

The foregoing two factors lend great significance to the hope for future implementation of MUXSIM results.

MUXSIM FUTURE

The future of MUXSIM is typified by the question, "Where should (or might) MUXSIM go from here?" In treating this question, some opinions on the matter are presented, together with a list of some areas where MUXSIM improvements or enhancements might be desirable. The latter are derived from experiences during the implementation, verification, and limited operational use of MUXSIM.

To review the situation briefly, MUXSIM has been designed, implemented, and verified, but has not yet been extensively tested through actual use. Therefore, it is felt that the next logical step is the operational evaluation of MUXSIM, and that such an evaluation will provide the best possible sources of inputs regarding needed improvements and enhancements, if any.

Hopefully, such an operational evaluation (Phase IV) will be conducted and the claims of MUXSIM cost-effectiveness will be fully justified by the results. With the continuing importance of multiplex systems in the design of modern aircraft, there are several ongoing Air Force development programs in which MUXSIM could be put to the test. It is felt that in the case of a tool such as MUXSIM, evolutionary improvements based solidly on its use history are the best kind. It is noteworthy that most successful, modern software simulation packages have been developed by this route; SIMSCRIPT,[2] GASP,[3] and GPSS[4] are cases in point.

Although actual use is generally the best source of information on MUXSIM development enhancements that may be desired, there are some areas where the need for further development is already quite clear. For example, a larger workload library is clearly desirable; in recognition of this fact, AFAL has already extended the MUXSIM effort to include additional workloads for use in defining multiplex systems for integrated digital avionics for other aircraft. This effort is soon to be completed and should immediately extend the range of useful results that can be expected from MUXSIM. Other specific possibilities for MUXSIM development which emerge as MUXSIM goes through the implementation and verification stages include the following:

- Extended MUXSIM dynamic model workloads
- Growth of dynamic models to allow detailed investigation of multiplex processing such as bus controller loading phenomena
- Further development of static models to allow study of other hardware configurations
- Incorporation of discrete event processor models which implement the processing and data transfer interactions

Many other possibilities could be mentioned in addition to the above; however, consistent with the view that future development needs are best established through real use experience, implementation of the above can only be recommended as each specific requirement becomes firm.

CONCLUSIONS

The two key design features, which are intended to make the tool simple to use and versatile, have already been established as successful:

- The coached Executive subsystem, which permits a person relatively unfamiliar with MUXSIM to actually use the system for familiarization and training and at the same time perform useful work.
- The modular construction, which provides versatility by allowing expansion of capability and/or direct replacement or substitution of algorithms to allow the user to model his specific system.

The operational evaluation phase that follows should put MUXSIM through a realistic test in a multiplex system design effort. Hopefully, such a test will reveal that MUXSIM is a truly useful and beneficial tool. Also, such a test is viewed as the best possible source of input for further MUXSIM development needs.

Hopes for using MUXSIM results in the design of multiplex systems are high because other programs related in nature which have been conducted in the past show that computer-aided design tools of this kind prove to be extremely useful. A high degree of optimism would thus seem to be justified concerning the value of MUXSIM as a computer-aided tool whose results are directly implementable.

REFERENCES

1. Pritsker, A. Alan B., "The GASP IV Simulation Language," Wiley-Interscience, New York, N.Y., 1974.
2. Markowitz, Harry M., Bernard Hausner, and Herbert W. Karr, "SIMSCRIPT - A Simulation Programming Language," Prentice-Hall, Englewood Cliffs, N.J., 1963.
3. Kiviat, P. J., "GASP - A General Activity Simulation Program," Monroeville, Pa.: Applied Research Laboratory, U.S. Steel Corp., July 8, 1963.
4. Gordon, Geoffrey, "A General Purpose Systems Simulation Program," Proc. of EJCC, Washington, D.C., New York: The Macmillan Co., 1961, 87-104.