NASA Technical Memorandum 87216

NASA-TM-87216 19860006751

Simulating a Small Turboshaft Engine in a Real-Time Multiprocessor Simulator (RTMPS) Environment

Edward J. Milner and Dale J. Arpasi
Lewis Research Center
Cleveland, Ohio

January 1986

NASA

SIMULATING A SMALL TURBOSHAFT ENGINE IN A REAL-TIME MULTIPROCESSOR SIMULATOR (RTMPS) ENVIRONMENT*

Edward J. Milner and Dale J. Arpasi
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

A Real-Time Multiprocessor Simulator (RTMPS) has been developed at NASA Lewis Research Center. The RTMPS uses parallel microprocessors to achieve computing speeds needed for real-time engine simulation. This report describes the use of the RTMPS system to simulate a small turboshaft engine. The process of programming the engine equations and distributing them over one, two, and four processors is discussed. Steady-state and transient results from the RTMPS simulation are compared with results from a main-frame-based simulation. Processor execution times and the associated execution time savings for the two and four processor cases are presented using actual data obtained from the RTMPS system. Included is a discussion of why the minimum achievable calculation time for the turboshaft engine model was attained using four processors. Finally, future enhancements to the RTMPS system are discussed including the development of a generalized partitioning algorithm to automatically distribute the system equations among the processors in optimum fashion.

INTRODUCTION

Modern jet engines and their controls continue to become more and more complex. Over the years, engine designs have continued to increase thrust-to-weight ratios. To gain this increased efficiency, the engines must operate closer to surge conditions - conditions which can lead to catastrophic engine stalls. Therefore, digital electronic engine controls are being developed that provide close monitoring of engine parameters and precise control of fuel flow and variable geometry.

Using a real-time simulation of an engine can facilitate the development of these digital controls because it allows the actual control hardware and software to be tested in a realistic fashion. If the real-time simulation is implemented on a digital computer, the time required by the computer to complete one pass through the equations representing the engine must not be greater than the integration step size. However, because of the massive amount of calculation required to model a complex engine, real-time simulation of modern jet engines on a single processor requires a fast, dedicated main-frame computer or simplification of the engine model.

A Real-Time Multiprocessor Simulator (RTMPS) has been developed at NASA Lewis Research Center to demonstrate the potential of using parallel microprocessors for real-time engine simulation (refs. 1 and 2). In addition to the RTMPS hardware, a structured, macro-based Real-Time Multiprocessor Language (RTMPL) (refs. 3 and 4) and a Real-Time Multiprocessor Operating System (RTMPOS) (refs. 5 and 6) were developed to form an RTMPS programming environment. The RTMPL software allows engineering-level personnel to develop simu-

*Material, not previously published, presented at the 1986 SCS Multiconference, sponsored by the Society for Computer Simulation, San Diego, California, January 23-24, 1986.

lation programs for the RTMPS. The operating system allows the user to interactively communicate with the RTMPS system

This report describes the use of the RTMPS system to simulate a small turboshaft engine. The turboshaft simulation represents the first real application of the RTMPS system. Therefore, the paper presents an evaluation of the RTMPS from a user's point of view.

Key features of the RTMPL and RTMPOS and their use to develop the turboshaft engine simulation are discussed and demonstrated. The process of programming the engine equations in RTMPL and distributing the RTMPL equations over one, two, and four processors is also discussed. Steady-state and transient results from the RTMPS simulation, obtained using the RTMPOS, are compared with results from a mainframe-based simulation. Processor execution times and the associated speed-up for the two and four processor cases are presented using actual data obtained from the RTMPS system. Included is a discussion of how the minimum achievable calculation time for the turboshaft engine model was attained using four processors. Finally, planned enhancements to the RTMPS system are discussed including the development of a generalized partitioning algorithm to automatically distribute the system equations among the processors in optimum fashion.

THE REAL-TIME MULTIPROCESSOR SIMULATOR HARDWARE CONFIGURATION

A schematic of the RTMPS hardware is shown in figure 1. The RTMPS consists of 2(n+1) processors (i.e., 16-bit, single-board microcomputers) 2n shared memories (i.e., dual-port memory boards) connected via two data busses. Two processors are reserved and the remaining 2n processors are available to operate on portions of the simulation. One reserved processor, designated the Front End Processor (FEP), is the communication link between the user and the simulation. The FEP allows the simulation to run interactively. The user may monitor the results coming from a simulation while it is running and may also make changes to key simulation parameters "on-the-fly" while a transient is being run. The other reserved processor, designated the Real-Time Extension (RTX), is the communication link between the simulator and external devices such as actuators or controllers. The FEP and the computational (COMP) processors, are linked through the Interactive Information Bus. The RTX and the remaining processors, called preprocessors (PREP), are linked through the Real-Time Information Bus. In general, both the COMP and PREP processors can be used for solving the simulation equations. For interactive simulation, only the Real-Time Information Bus is used to transfer real-time data between the processors. This is because the Interactive Information Bus may be busy servicing user requests.

In each simulation channel, the COMP and PREP processors communicate through the shared memory. If data are to be transferred between two COMP processors, the data must be sent from the first COMP processor, through shared memory, to its corresponding PREP processor, across the Real-Time Information Bus to the PREP processor of the channel to receive the data, then finally through that channel's shared memory to the COMP processor requiring the information. Because of this round-about path, transfers of data between COMP processors are avoided when formulating an RTMPS simulation.

THE REAL-TIME MULTIPROCESSOR LANGUAGE (RTMPL)

RTMPL ia a structured, high-order language designed to facilitate the development of error-free, time-efficient simulations in a digital multiprocessor environment (refs. 3 and 4). It is a macro-based language which offers the advantage that any simulation written in the language is independent of the processors on which it is to be executed. Simulation source code executed on one set of processors need not be changed to be able to execute on another set of processors. Only the macro-based language operations need be retargeted for the new machine. An RTMPL language utility (translator) has been developed that converts the RTMPL source code to a list of macro calls.

An assembly language programmer performs a one-time systems task to develop the optimum assembly code for the language operation macros. The execution of the simulation is only as efficient as the pre-programmed macros.

For each processor (i.e., program) in the simulation, the user specifies each constant and variable used in the program and their data type, precision, scale factor, and value, as appropriate. The language utility keeps track of the constants and variables associated with each processor, which values must be sent from one processor to another, when a current value of each has been calculated so that data transfer can begin, and when all processors have completed the calculations assigned to them.

A small turboshaft engine simulation will be used later to further illustrate features of RTMPL.

THE REAL-TIME MULTIPROCESSOR OPERATING SYSTEM (RTMPOS)

RTMPOS provides the user with a versatile, interactive means for controlling and obtaining results from a simulation executing on the RTMPS. RTMPOS resides on the Front-End-Processor (FEP) and serves as the interface between the user and the simulator. It allows him to load, run, debug, and obtain and/or monitor results from the simulation. The user may also use RTMPOS to modify and specify the computational flow of the simulation while it is executing.

The RTMPL translator produces "data base" files that contain information needed for that interactive execution of the simulation. The RTMPOS uses the data base information at run-time to allow the user to interactively execute the simulation. RTMPOS also allows the data base to be edited; that is, it allows changing/displaying values of constants and initial conditions on variables at run time. This modified data base may then be saved on disk, if desired, and any portion or all of it may be listed on a printer.

At run time the program control features of RTMPOS are used to load the program modules from the disk files into the desired simulation channels and to activate the execution of each of the program modules. Just as with an analog or hybrid computer, three modes of program execution are available through RTMPOS. They are: RUN, HOLD, and STOP. In the RUN mode all programs are activated on the simulator and they run repeated update cycles of the simulation until the user issues a STOP command or the simulator issues a halt advisory. (A halt advisory results from an error condition in the simulator and terminates the simulation execution.) In the HOLD mode, user specified

variables in any or all program modules are held at current or preselected values for a specified number of update cycles as requested by the user. In the STOP mode, simulation execution is suspended and the simulator is halted.

A very useful feature of the RTMPOS is the creation of a permanent session history disk file. This file saves all user commands, prompts, and messages from the operating system during a user session on the simulator. Hence, the entire session is self-documenting and the user can review the sequence of work performed in the session. Successive session histories document the complete development and debug process during the buildup of a complex simulation so the user has a record of each step in the simulation development process. Furthermore, the commands in a session history file can be automatically read and executed by RTMPOS, allowing the user to quickly return to a condition obtained in a previous session.

Selected simulation parameters may also be sampled as a function of time during any transient run of a simulation. These data are saved in a disk file and they may be listed in tabular form or they may be transferred to a graphics utility where they may be plotted to display functional relationships in visual form.

Features of RTMPOS will be further illustrated using a simulation of the small turboshaft engine model discussed in the next section.

THE SMALL TURBOSHAFT ENGINE MODEL

The process of testing and debugging the Real-Time Multiprocessor Simulator (RTMPS) hardware and system software required the selection of a suitable dynamic simulation model. It was decided to use an available aircraft engine model. That model was felt to be representative of modern air-breathing engines, yet simple enough to allow concentrating on the RTMPS system hardware and software to be debugged. Simulating a more complex engine model would hinder efficient debugging of the RTMPS hardware and software.

Selected was a model of a small turboshaft engine in the 20 000 lb thrust class. Included in the model are mathematical representations of the following: a single-centrifugal-stage, five-axial-stage compressor that includes variable inlet guide vanes and variable stator vanes for the first two stages; an annular combustor; a two-stage, axial, air-cooled gas generator turbine that drives the compressor rotor; compressor exit bleed air that cools the turbine; a second two-stage turbine that is uncooled and that has a coaxial drive shaft that extends forward through the gas generator turbine and connects to the engine output shaft.

A digital computer simulation of the small turboshaft engine model had been developed previously on a central, mainframe computer. The simulation had been thoroughly debugged and was operational. In addition, a large base of data from the simulation was on file. This information could be used to verify results coming from the RTMPS.

As will be shown later, this small turboshaft engine model turned out to be nearly ideal for verifying RTMPS operation. It was of sufficient complexity to offer a challenging partitioning problem for the processors. In the process of determining what equations should go on the various processors, much insight

was gained and generic partitioning algorithms were developed. Since the turboshaft engine model only had six dynamic state variables, the simulation of this engine model did not produce long execution times and bog down the checkout of the RTMPS system.

For the purpose of checking out the RTMPS, results coming from the mainframe simulation of the small turboshaft engine model were used as the standard against which the RTMPS results would be judged.

RESULTS AND DISCUSSION

In the process of checking out the RTMPS hardware and software, three RTMPS simulations of the small turboshaft engine were developed: a single-processor simulation, a dual-processor simulation, and a quad-processor simulation. The purpose of the single-processor simulation was to check the basic hardware and software functions and to develop a data base of results from the RTMPS to be used as a basis for further checkout of the system. The dual-processor simulation exercised some of the interface software and hardware including the ability of a computational processor and a preprocessor to exchange information through a shared memory in a timely and effective fashion. The quad-processor simulation further exercised the interface software/hardware system and demonstrated the effectiveness of the dual-bus, parallel-processing concept. The objective was to show that the RTMPS, together with the RTMPL high-order multiprocessor language and RTMPOS user operating system, provided an effective and efficient environment for real-time simulation.

The Single-Processor Simulation

One of the first steps in developing an RTMPL simulation is the definition of constants and variables. RTMPL is a structured language and, as such, requires that all constants and variables used in the simulation be thoroughly defined. The language allows no numeric constants to be used in the simulation equations. Hence, each constant must be assigned an alphanumeric name in the constant definition section of the program. A portion of the constant definitions for the turboshaft engine model is shown in figure 2.

The definition section begins with the declaration CONSTANT. Following this declaration is a list of the constants used in the program. Notice that, for each constant, its name is immediately followed by its type and precision. The type may be either integer (I) or scaled fraction (S). Integers are counting numbers and do not have a decimal point. Scaled fractions are fixed point numbers with an absolute value less than one. The precision is specified by either 1, 2, or 3 with 1 signifying single precision and 2 and 3 specifying double and triple precision, respectively. Scaled fractions require a further specification, namely, a scale factor. Because the scaled fraction is limited to values between -1 and +1, a scaled fraction is obtained by dividing the raw data value by some appropriate constant value called the scale factor. RTMPL requires that the scale factor be a power of 2. So for a scaled fraction, after the data type and precision, in the definition declaration, is a slash (/) followed by the exponent of the power of 2 scale factor. Integers are not scaled, so no scale factor exponent appears in their definition. Finally, within brackets appears the value of the constant. If a data table is being

defined, the number of values making up the table is followed by brackets holding the table values.

The definition for variables in the program follows basically the same format. A portion of the variable definitions for the small turboshaft simulation is shown in figure 3. The definition section begins with the declaration VARIABLE. Following this declaration is a list of the variables used in the program. For each variable, its name is immediately followed by its type, its precision, and its scale factor exponent, if the variable is a scaled fraction. This is exactly the same as was done for the definition of the program constants. Within brackets for variables, however, are not only the variable initial condition value, but also what is designated as a hold value. This is the value that the variable will be set to by the RTMPOS if the variable is chosen to be "held fixed" or "frozen" under the RTMPOS HOLD mode of program execution. In figure 3, the slash (/) within the brackets signifies that the number following is both the initial condition value and the hold value for that variable. (A separate hold value for the variable would be specified by a number preceding the slash.)

The equations defining the small turboshaft engine model for this simulation are contained in an executive section, a portion of which is shown in figure 4. This section begins with the declaration EXEC, followed by the executive name. Following this declaration is the collection of equations making up the simulation. The operators used in these equations are defined RTMPL operations and the program appearing in the executive section will not change, no matter what computer system it is to be executed on. What will change are the system macros which translate each RTMPL operation used in the simulation into corresponding assembly code. RTMPL translates the EXEC code, as input by the user, into assembly language source code corresponding to the computer system to be used to execute the program. In addition to the assembly language code, RTMPL automatically creates extensive documentation for the user. Included in the documentation are references giving the relative location of each constant and variable used in the program. The assembly language source code file can then be assembled and linked to obtain an object file of the program which can be executed on the target computer system. A file containing the complete assembly code listing of the program to be executed is created in the process. This file can be printed out to obtain a hard copy record of the contents of each memory location used by the program.

The small turboshaft engine simulation was executed on the single processor RTMPS at four typical steady-state operating conditions. Steady-state results agreed very well with the steady-state results from the mainframe simulation standard at the corresponding operating conditions. Typical results are shown in table I where selected values at one of the steady-state conditions are shown for the two simulations. The RTMPS results agree with the mainframe standard to within 0.2 percent. These small errors occur mainly because the RTMPS simulation uses scaled fraction arithmetic whereas the mainframe simulation uses floating-point arithmetic. Also, the RTMPS simulation is run on a Motorola Exormacs development system based on the M68000 microprocessor chip with 16-bit memory. The mainframe computer was an IBM 370 with 32-bit memory.

Because of the excellent match between steady-state results from the RTMPS and the mainframe standard, it was felt that the RTMPS hardware and software were operating correctly. Also, RTMPL was proving itself to be an extremely effective, high-order language. A careful examination of the program code

revealed that it was producing assembly code that was 99 percent as efficient as code produced by an experienced assembly language programmer. RTMPOS also was proving itself to be a valuable tool. RTMPOS properly loaded the simulation into the RTMPS, and allowed the user to monitor key parameters as the simulation was achieving steady-state conditions.

A "stale data" error occurs when data from different calculation cycles are mixed - for example, when a parameter value not corresponding to the current calculation cycle is used to update parameters in the current cycle. Subtle differences caused by these kinds of errors can mistakenly be attributed to roundoff. To flag these, and other errors, if they existed, in comparing single, dual, and quad-processor simulation results, engineering unit conversions of the RTMPS memory contents were purposely printed in their entirety using the RTMPOS. Printed in their entirety, corresponding engineering unit conversions were expected to match exactly. The automatic unscaling of simulation parameters by the RTMPOS was an extremely welcome feature of the operating system. The RTMPOS further allowed the user to change operating conditions easily, and created a session history file in which to store the results.

To examine the dynamic characteristics of the single-processor RTMPS simulation and to check the data sampling and related features of the RTMPOS, a 40 sec transient was executed, starting at one of the four selected steady-state conditions. The first 2 sec was a steadying out period, allowing the simulation to reach a true steady-state condition. Then, during the next 7 sec, engine fuel flow was ramped from its design value to a value of 0.784 of design, which was held for an additional 18 sec. This was to allow the transient response to settle out so that the simulation could again approach a steady-state condition. Engine fuel flow was then ramped back up to its design value over the next 4 sec and that value was held for the remaining 9 sec of the transient.

Figure 5 shows a graph of engine fuel flow versus time. Figures 6 to 8 show the corresponding time responses of gas generator turbine speed, compressor discharge temperature, and gas generator turbine inlet pressure, respectively. The agreement shown in figures 6 to 8 is representative of the agreement found for other parameters in the simulation. The RTMPS transient results agreed with the mainframe transient results to within 0.5 percent.

The Dual-Processor Simulation

As shown in the last section, the RTMPS single-processor results were in excellent agreement with corresponding output from the mainframe standard. This demonstrated the validity of the RTMPS single-processor simulation and provided a basis for checking multiprocessor simulations of the small turboshaft engine on the RTMPS.

When using a multiprocessing system, the simulation and its results should remain unchanged; only the effective calculation time should be decreased. Hence, if the single-processor equations are distributed among two or more processors, the output data from the resulting simulation should agree <u>exactly</u> with corresponding output from the single-processor simulation. Without <u>exact</u> agreement, the system hardware and/or software would be judged to be not working properly.

To partition the simulation equations for solution on a multiprocessor system, one must consider the execution times for each equation in the simulation. The execution time for each equation in the single-processor RTMPS simulation was obtained by summing estimated execution times for each operation required to solve the equation. Hence, each equation then had an execution time associated with it. Given this timing information, the small turboshaft engine simulation equations could be divided to run on multiple processors thereby reducing the effective calculation time. The following criteria were used to evaluate various partitioning schemes: 1) the time to complete the calculations on each processor should be closely balanced; 2) processor use should be maximized; that is, idle time waiting for data from other processors should be kept to a minimum; and 3) the number of data transfers between processors should also be kept to a minimum.

For the dual-processor case, the simulation equations were split as shown in figure 9. The resulting calculation times for the processors were within a few clock cycles of being equal. Neither processor wasted time waiting for data from the other. And only nine transfers of data between processors were required.

The nine data transfers between processors exercised the shared memory between processors. When this simulation was first run on the dual-processor RTMPS with the same transient used for the single-processor simulation, the results did not agree with those coming from the single-processor simulation. Several system software bugs and a timing flag error were discovered and corrected before exact duplication of the single-processor results was achieved. Without the capability of printing the engineering unit conversions in their entirety, determining whether the two simulations agreed exactly would have been a very time consuming task. It would have required accessing memory and examining its hexadecimal contents to make sure that the simulation results were the same. The operating system's ability to automatically convert these hexadecimal numbers to very accurate equivalent engineering units allowed this effort to be readily accomplished. It proved to be a particularly useful tool in tracing the almost negligible differences between corresponding single and dual-processor simulation outputs occurring midtransient. The criterion for exact duplication of results, and being satisfied with nothing less, sparked a long, persistent effort which led to the discovery and correction of the subtle timing flag problem.

A portion of the steady-state output from the single and dual-processor simulations is shown in table II. The corresponding results matched exactly. Figures 10 to 12 show comparisons of time transients of gas generator turbine speed, compressor discharge temperature, and gas generator turbine inlet pressure, respectively, for the single and dual-processor simulations. The corresponding transients were exact duplicates for the two simulations.

Satisfied that the hardware and software were working for this dual-processor case, further partitioning and distribution of the small engine simulation equations could be pursued with confidence.

The Quad-Processor Simulation

To partition the simulation equations for the quad-processor RTMPS, it was useful to compute "can start" and "can end" times for each equation in the

single-processor simulation. The earliest that the calculation of an equation "can start" is that time at which the calculation of every parameter to the right of the equal sign has been completed. The earliest the calculation of an equation "can end" is its "can start" time plus its execution time. Equations with only constants or state variables (outputs of integrators) to the right of the equal sign "can start" at time zero. In turn, subsequent equations (which depend on results from previous equations) "can start" at the latest "can end" time of the parameters to the right of the equal sign. For any simulation this information can be obtained using this purely mechanical method. For the small turboshaft engine simulation, the information is shown in figure 13.

The equation string culminating with the equation with the latest "can end" time was placed on a processor by itself. The other equations were divided among three other processors in a way such that the total execution time of none of the three exceeded that of the first. The resulting distribution of the equations among the four processors is shown in figure 14. Results obtained with the quad-processor RTMPS matched exactly with the single and dual-processor results. Time transients of gas generator turbine speed, compressor discharge temperature, and gas generator turbine inlet pressure are shown in figures 15 to 17, respectively. The transients shown in these figures are exact duplicates of the corresponding single and dual-processor transients shown in figures 10 to 12. A portion of the steady-state output for the single, dual, and quad-processor simulations is shown in table III. The steady-state data were identical. Results were not affected by running the simulation on one processor, two processors, or four. Only the effective calculation time was affected.

Effective Calculation Time

As was alluded to in the introduction section of this report, the reason for developing a multiprocessor simulator is to take advantage of any parallelism in the simulation model for the purpose of decreasing the effective calculation time. The single-processor simulation required 3.83 msec to complete one pass through all the calculations. The dual-processor simulation required 2.26 msec and the quad-processor simulation required 1.81 msec.

Thus, the dual-processor configuration executed the small turboshaft engine simulation 1.7 times faster than did the single-processor configuration. Ideally, all else being equal, one could expect two processors to complete a given set of calculations twice as fast as if only one processor were available. Practically, this is not true. Some time is lost in multiprocessor overhead, primarily because of data transfer requirements between processors. Data calculated on one processor and needed by another requires that the RTMPS transfer that data over the Real-Time Information Bus and/or through shared memory. These transfer operations add to the effective calculation time. Also, the simulation equations may not partition equally between the two processors. The effective calculation time will be dictated by the "critical path" time. This latter effect will be discussed in the next section.

The data transfer time among processors is even more pronounced in the quad-processor configuration. Ideally, four processors would complete a set of calculations four times faster than only one processor. But for the RTMPS, four processors execute the small turboshaft simulation 2.1 times as fast as

the single-processor configuration. The "critical path" execution time, which is 1.81 ms, is the dominant factor, as will be discussed in the next section.

The RTMPS multiprocessor system used for this study had relatively slow 16-bit processors by today's standards. Using, newer 32-bit microcomputer boards will increase the processing speed by a factor of four or five. But the existing system did demonstrate the effective time savings that can be realized by taking advantage of parallelism present in a simulation model.

Partitioning and Effective Time Savings Limits

The "critical path" of a simulation consists of the serial string of calculations which requires the longest execution time. For the quad-processor simulation of the small turboshaft engine model, the "critical path" resided on processor B in figure 14. The last variable calculated on that processor, NG, had the longest "can end" time as discussed in the Dual Processor Simulation section. Also, for each equation, the variable appearing to the left of the equal sign was required as an argument in the following equation. On an equation basis, the equation set residing on processor B could not be distributed further among processors so as to be made to execute faster. If any of the other three processors in the quad-processor simulation had a longer execution time than processor B, using additional processors to take advantage of any further parallelism in the simulation model would have an effect on the effective calculation time. This is so because the "critical path" execution time is the minimum time within which the simulation can be executed. In the case of the small turboshaft engine model, though, this was not the situation.

Timing studies showed that using four processors to simulate the small turboshaft engine model yielded an effective speed up of 2.1 over using only a single processor. As previously stated, this was the fastest that the turboshaft engine model could be run on the RTMPS system using equation level partitioning. Using more processors would have no further effect at reducing the calculation time since the "critical path" of the simulation model was already on one processor by itself, and the execution time of no other processor was greater than this one.

Partitioning of simulation equations involves a number of straight forward tasks. However, they are rather time consuming to carry out by hand, as was done for this study. To automate these tasks, development of a general partitioning algorithm has been undertaken at the Lewis Research Center. The partitioning algorithm will use timing information from the RTMPL translator to determine "can start" and "can end" times for each equation and will distribute the equations among the available processors in optimum fashion. Details will be presented in a forthcoming NASA report.

CONCLUDING REMARKS

A small turboshaft engine has been modeled and simulated on the real-time multiprocessor digital simulator (RTMPS), developed at NASA Lewis. A general purpose, macro-based, high-order, multiprocessor language, RTMPL, was used to program the RTMPS. An interactive multiprocessor operating system, RTMPOS, was used to load the simulation programs and to control the operation of the RTMPS. The RTMPS hardware and the RTMPL and RTMPOS software had previously been

developed at NASA Lewis. The RTMPS hardware and software proved to be useful, reliable tools for simulation.

The results of this study demonstrated that the RTMPS can take advantage of parallelism in a simulation model and significantly reduce the effective calculation time for the simulation. This result indicates that a digital multiprocessor simulator such as the RTMPS can be a viable alternative to large mainframe digital and analog/hybrid computers for achieving real-time simulation of complex dynamic systems. In the current study, involving a turboshaft engine model, an effective time savings factor of 2.1 was realized. Four processors were used simultaneously to evaluate the simulation equations. Other dynamic system models, containing more parallelism in the equations, would benefit from using additional processors and would experience a greater speedup. Partitioning at a level lower than the equation level (at the operation level, for instance) might also improve performance. The use of state-of-the-art processors and busses is expected to result in simulators that can provide real-time simulation speed for a wide range of simulation applications.

REFERENCES

1. Blech, R.A.; and Arpasi, D.J.: An Approach to Real-Time Simulation Using Parallel Processing. Proceedings of the 1981 Summer Computer Simulation Conference, AFIPS Press, Arlington, 1981, pp. 355-360.

2. Blech, R.A.; and Arpasi, D.J.: Hardware for a Real-Time Multiprocessor Simulator. Distributed Simulation 1985, P. Reynolds, ed., Society for Computer Simulation, La Jolla, 1985, pp. 43-52.

3. Arpasi, D.J.: RTMPL - A Structured Programming and Documentation Utility for Real-Time Multiprocessor Simulations, NASA TM-83606, 1984. (also published in the, Proceedings of the 1984 Summer Computer Simulation Conference)

4. Arpasi, D.J.: Real-Time Multiprocessor Programming Language (RTMPL): Users Manual, NASA TP-2422, 1985.

5. Cole, G.L.; Operating System for a Real-Time Multiprocessor Propulsion System Simulator, NASA TM-83605, 1984. (also published in the, Proceedings of the 1984 Summer Computer Simulation Conference)

6. Cole, G.L.; Operating System for a Real-Time Multiprocessor Propulsion System Simulator.^ Users Manual. NASA TP-2426, 1985.

TABLE I. - COMPARISON OF SINGLE-PROCESSOR AND MAIN-FRAME COMPUTER STEADY-STATE RESULTS

SINGLE PROCESSOR	MAINFRAME
NG = 41522.	NG = 41516.
NP = 21194.	NP = 21163.
P41 = 173.18	P41 = 173.14
P45 = 37.133	P45 = 37.116
T3 = 1164.0	T3 = 1164.1
WS3 = .16321	WS3 = .16314

TABLE II. - COMPARISON OF SINGLE-PROCESSOR AND DUAL-PROCESSOR STEADY-STATE RESULTS

SINGLE PROCESSOR RESULTS

```
STSE1CH2.C.DHQTH4 (LV) =  3.98378906250000000E+001
STSE1CH2.C.DHQTH5 (LV) =  2.26777343750000000E+001
STSE1CH2.C.DH41 (LV) =  1.59328125000000000E+002
STSE1CH2.C.DH45 (LV) =  6.50507812500000000E+001
STSE1CH2.C.FAR41 (LV) =  1.61914825439453125E-002
STSE1CH2.C.H2 (LV) =  1.21375000000000000E+002
STSE1CH2.C.H25 (LV) =  1.90609375000000000E+002
STSE1CH2.C.H3 (LV) =  2.68401567220687870E+002
STSE1CH2.C.H41 (LV) =  5.54281250000000000E+002
STSE1CH2.C.H44 (LV) =  3.94968750000000000E+002
STSE1CH2.C.H45 (LV) =  3.80062500000000000E+002
STSE1CH2.C.H49 (LV) =  3.15031250000000000E+002
```

DUAL PROCESSOR RESULTS

```
STSE2CH2.C.DHQTH4 (LV) =  3.98378906250000000E+001
STSE2CH2.C.DHQTH5 (LV) =  2.26777343750000000E+001
STSE2CH2.C.DH41 (LV) =  1.59328125000000000E+002
STSE2CH2.C.DH45 (LV) =  6.50507812500000000E+001
STSE2CH2.C.FAR41 (LV) =  1.61914825439453125E-002
STSE2CH2.P.H2 (LV) =  1.21375000000000000E+002
STSE2CH2.P.H25 (LV) =  1.90609375000000000E+002
STSE2CH2.P.H3 (LV) =  2.68401567220687870E+002
STSE2CH2.C.H41 (LV) =  5.54281250000000000E+002
STSE2CH2.C.H44 (LV) =  3.94968750000000000E+002
STSE2CH2.C.H45 (LV) =  3.80062500000000000E+002
STSE2CH2.C.H49 (LV) =  3.15031250000000000E+002
```

TABLE III. - COMPARISON OF SINGLE-PROCESSOR, DUAL-PROCESSOR,

AND QUAD-PROCESSOR STEADY-STATE RESULTS

SINGLE PROCESSOR RESULTS

```
STSE1CH2.C.THTA41 (LV) =  4.10742187500000000E+000
STSE1CH2.C.THTA45 (LV) =  2.94198630005121230E+000
STSE1CH2.C.TORQC (LV) =  2.08301496505737300E+002
STSE1CH2.C.TORQ41 (LV) =  2.07286296844482420E+002
STSE1CH2.C.TORQ45 (LV) =  1.90781534194946290E+002
STSE1CH2.C.T25 (LV) =  8.12625000000000000E+002
STSE1CH2.C.T25Q2 (LV) =  1.60015869140625000E+000
STSE1CH2.C.T3C (LV) =  1.13496919250488280E+003
```

DUAL PROCESSOR RESULTS

```
STSE2CH2.C.THTA41 (LV) =  4.10742187500000000E+000
STSE2CH2.C.THTA45 (LV) =  2.94198630005121230E+000
STSE2CH2.P.TORQC (LV) =  2.08301496505737300E+002
STSE2CH2.C.TORQ41 (LV) =  2.07286296844482420E+002
STSE2CH2.C.TORQ45 (LV) =  1.90781534194946290E+002
STSE2CH2.P.T25 (LV) =  8.12625000000000000E+002
STSE2CH2.P.T25Q2 (LV) =  1.60015869140625000E+000
STSE2CH2.P.T3C (LV) =  1.13496919250488280E+003
```

QUAD PROCESSOR RESULTS

```
STSE4CH3.C.THTA41 (LV) =  4.10742187500000000E+000
STSE4CH3.C.THTA45 (LV) =  2.94198630005121230E+000
STSE4CH2.P.TORQC (LV) =  2.08301496505737300E+002
STSE4CH3.P.TORQ41 (LV) =  2.07286296844482420E+002
STSE4CH3.C.TORQ45 (LV) =  1.90781534194946290E+002
STSE4CH2.C.T25 (LV) =  8.12625000000000000E+002
STSE4CH2.C.T25Q2 (LV) =  1.60015869140625000E+000
STSE4CH2.C.T3C (LV) =  1.13496919250488280E+003
```

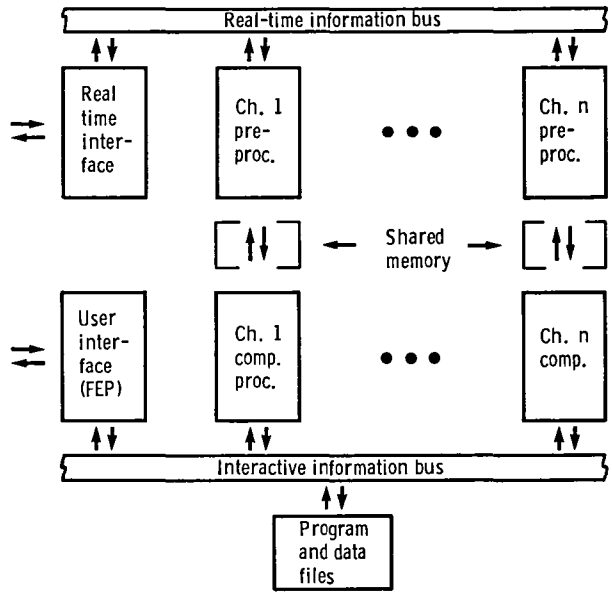


Figure 1. - Real-time multiprocessor simulator (RTMPS) hardware configuration.

```
CONSTANT:

KDHB = S1/15,[18210.];
KRQV3 = S1/0,[.97174];
SWA31 = S1/5,[27.923];
PSTD = S1/4,[14.696];
TSTD = S1/10,[518.67];
SPST3 = S1/1,[.956];
SNDES = S1/9,[447.];
S0025 = S1/-3,[.0025];
S84 = S2/9,[8.4];
SW3T3 = S1/-2,[.2496];
S308 = S1/12,[308.];
S3298 = S1/2,[3.298];
S0856 = S2/4,[.0856];
STH41 = S1/-8,[.00183263];
KWGTRB = S1/-3,[.0876];
KV41 = S1/3,[6.17];
STRQ41 = S1/13,[7429.35];
S039 = S1/-4,[.039];
S115 = S2/1,[1.15];
S240 = S1/-2,[.240];
S239 = S1/-2,[.239];
STORQC = S1/13,[7429.35];
SNGDT = S1/8,[214.59];
S9623 = S1/0,[.9623];
XW2CB = S1/4,4[0.0,3.0,4.98,12.0];
NW2CB = I1,2[2,8];
ZB2 = S1/-1,4[.01057,.01057,.0090,.0090];
XW2C = S1/4,4[3.0,8.3,10.0,12.0];
NW2C = I1,2[2,8];
ZWXQ2 = S1/-3,4[.0851,.0846,.0815,.0779];
```

Figure 2. - Portion of RTMPL constant declaration section.

```
VARIABLE;

DEL2 = S1/1,C/.95795];
RTTH2 = S1/1,C/.98957];
WS3 = S2/-2,C/.16314];
T3 = S2/11,C/1164.1];
P3 = S2/9,C/184.54];
PS3 = S2/10,C/176.42];
PS3Q2 = S1/5,C/12.532];
T3Q2 = S1/2,C/2.2919];
T3C = S2/11,C/1164.1];
*** DERIVATIVE T3 ***;
T3DT = S2/10,2C/0.];
NG = S2/16,C/41516.];
NGC = S2/16,C/41953.];
PCNGC = S1/7,C/93.856];
WA2C = S1/5,C/8.2345];
WA2 = S2/4,C/7.9714];
B1 = S1/-1,C/0.];
B2 = S1/-1,C/.009];
WB25 = S2/3,C/.0717422];
WA3 = S2/4,C/7.8996];
WXQ2 = S1/-3,C/.0846061];
WB3 = S2/4,C/.69435];
P41 = S2/9,C/173.14];
TERM1 = S2/5,C/11.40];
WA31 = S1/4,C/7.2051];
*** DERIVATIVE WS3 ***;
WS3DT = S2/4,2C/0.];
H3 = S2/9,C/282.15];
FAR41 = S1/-4,C/.0180428];
H41 = S1/10,C/599.88];
```

Figure 3. - Portion of RTMPL variable declaration section.

```
EXEC; STFEEXEC[0];

*** ENGINE DYNAMICS ***;
DEL2 = P2/PSTD;
RTTH2 = SQRT(T2/TSTD);
P3 = KRQV3*WS3*T3;
PS3 = SPST3*P3;
PS3Q2 = PS3/P2;
T3Q2 = FUN1[XPRC1,NPRC1,ZTRC,PS3Q2];
T3C = T3Q2*T2;
NGC = NG/RTTH2;
PCNGC = NGC/SNDES;
WA2C = MAP[XPRC,YPNGC,NPRC,ZW2C,PS3Q2,PCNGC];
WA2 = (WA2C/RTTH2)*DEL2;
B1 = FUN1[XPNGC,NPNGC,ZB1,PCNGC];
B2 = FUN1[XW2CB,NW2CB,ZB2,WA2C];
WB25 = (B1+B2)*WA2;
WA3 = WA2-WB25;
WXQ2 = FUN1[XW2C,NW2C,ZWXQ2,WA2C];
WB3 = (WXQ2+S0025)*WA2;
WA31 = SQRT(SWA31*WS3*(P3-P41));
H3 = SH3T3*T3-S84;
FAR41 = STFE1CH1.P.WF/WA31;
H41 = (H3+KDHB*FAR41)/(S1+FAR41);
T41 = S3298*H41+S308;
THTA41 = STH41*T41+S0856;
W41 = KWGTR8*(P41/SQRT(THTA41));
PR45Q1 = P45/P41;
DHQTH4 = FUN1[XPRGT,NPRGT,ZDHGTQ,PR45Q1];
DH41 = DHQTH4*THTA41;
TORQ41 = ((STRQ41*DH41)/NG)*W41;
T25Q2 = S115+S039*PS3Q2;
```

Figure 4. - Portion of equations defining simulation model.

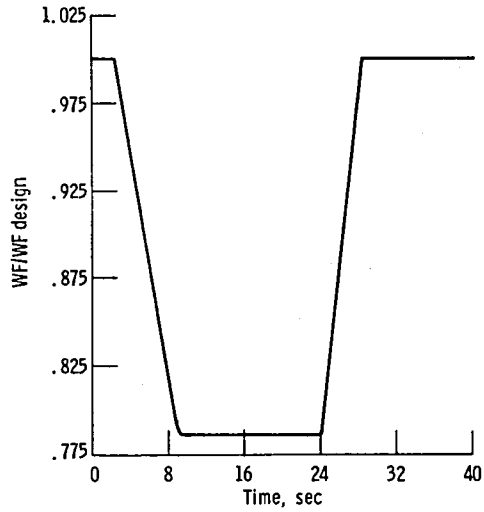


Figure 5. - Engine fuel flow versus time.

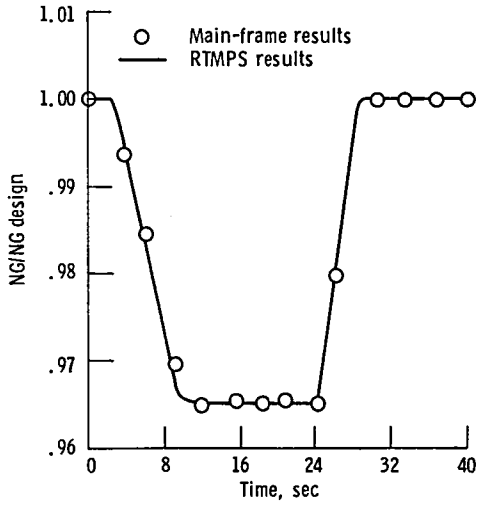


Figure 6. - Gas generator turbine speed versus time. Single processor.

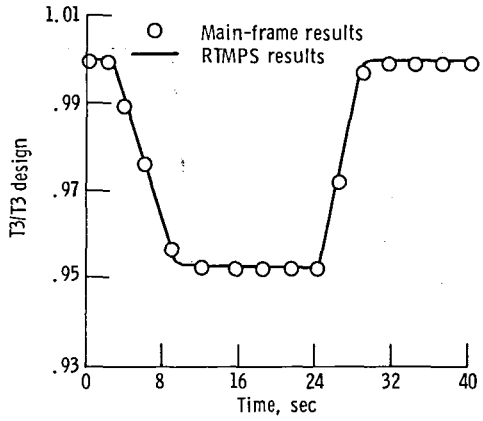


Figure 7. - Compressor discharge temperature versus time. Single processor.

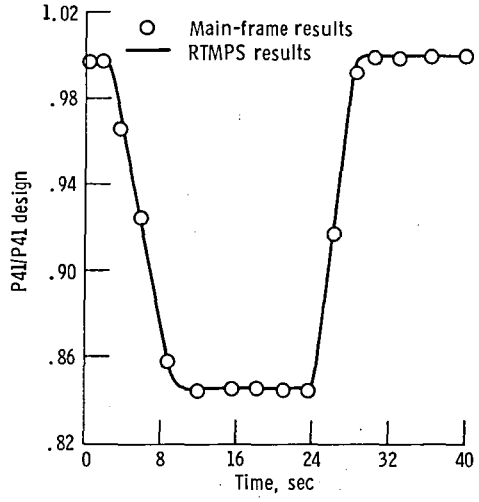
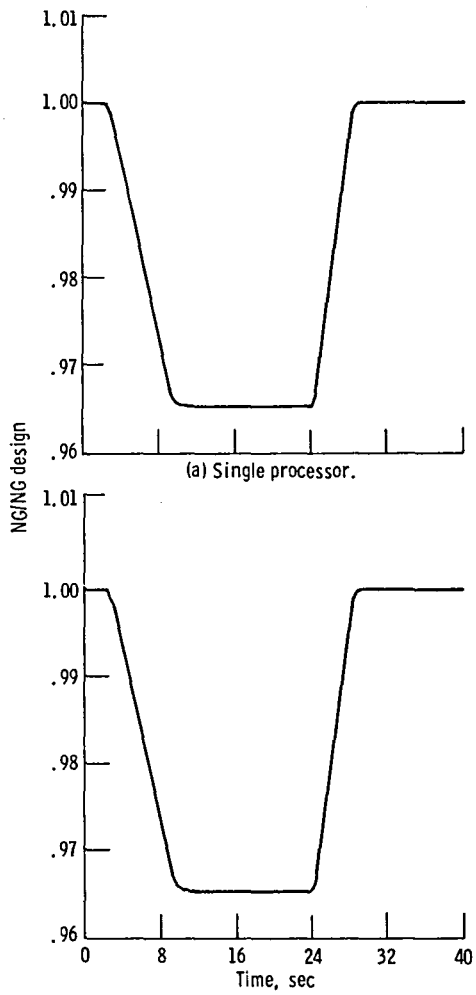


Figure 8. - Gas generator turbine inlet pressure versus time. Single processor.

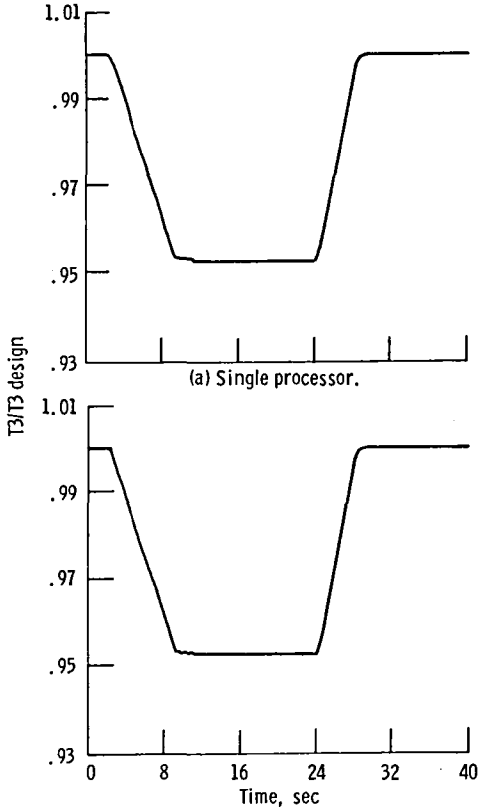
```
*** PROCESSOR A ***;

PR45Q1 = P45/P41;
*** VERIFY P3 ***;
*** VERIFY WS3 ***;
WA31 = SQRT(SWA31*STSE2CH2.P.WS3$1*(STSE2CH2.P.P3-P41));
FAR41 = STSE2CH1.P.WF/WA31;
*** VERIFY H3 ***;
H41 = (STSE2CH2.P.H3+KDHB*FAR41)/(S1+FAR41);
T41 = S3290*H41+S309;
THTA41 = STH41*T41+S0856;
W41 = KWGTRB*(P41/SQRT(THTA41));
DHQTH4 = FUN1CXPRGT,NPRGT,ZDHGTQ,PR45Q1];
DH41 = DHQTH4*THTA41;


*** PRECESSOR B ***;

P3 = KRQV3*WS3*T3;
H3 = SH3T3*T3-S84;
PS3 = SPST3*P3;
PS3Q2 = PS3/P2;
RTTH2 = SQRT(T2/TSTD);
NGC = NG/RTTH2;
PCNGC = NGC/SNDES;
WA2C = MAPCXPRC,YPNGC,NPRC,ZW2C,PS3Q2,PCNGC];
DEL2 = P2/PSTD;
WA2 = (WA2C/RTTH2)*DEL2;
*** VERIFY WXQ2 ***;
WB3 = (STSE2CH2.C.WXQ2+S0025)*WA2;
T25Q2 = S115+S039*PS3Q2;
```

Figure 9. - Portion of dual processor equation split.



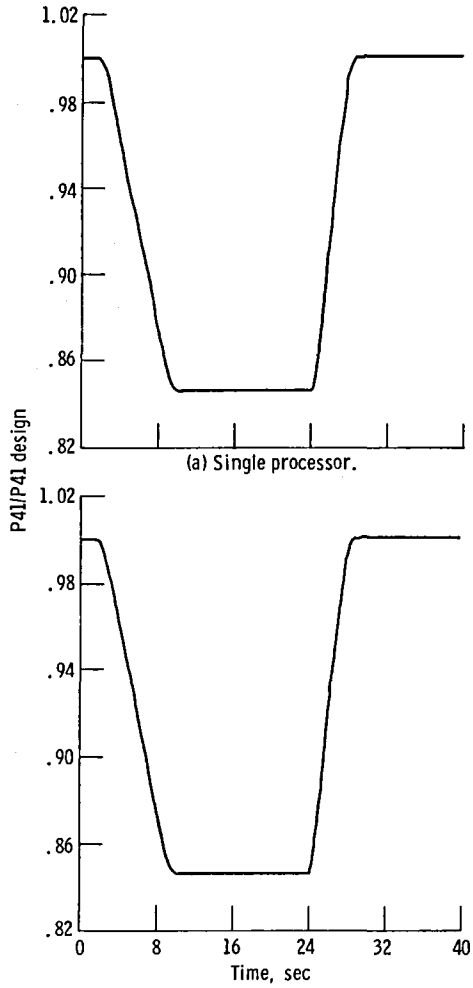
(b) Dual processor.

Figure 10. - Gas generator turbine speed versus time.



(b) Dual processor.

Figure 11. - Compressor discharge temperature versus time.



(b) Dual processor.

Figure 12. - Gas generator turbine inlet pressure versus time.

STMT TIME		"CAN START"	"CAN END"
218	DEL2 = P2/PSTD;	0	218
770	RTTH2 = SQRT(T2/TSTD);	0	770
214	WF = WFPH/SSEC;	0	214
246	P3 = KRQV3*WS3*T3;	0	246
122	PS3 = SPST3*P3;	246	368
204	PS3Q2 = PS3/P2;	368	572
598	T3Q2 = FUN1CXPRC1,NPRC1,ZTRC,PS3Q2];	572	1170
132	T3C = T3Q2*T2;	1170	1302
434	NGC = NG/RTTH2;	770	1204
198	PCNGC = NGC/SNDES;	1204	1402
2136	WA2C = MAPCXPRC,YPNGC,NPRC,ZW2C,PS3Q2,PCNGC];	1402	3538
292	WA2 = (WA2C/RTTH2)*DEL2;	3538	3830
598	B1 = FUN1CXPNGC,NPNGC,ZB1,PCNGC];	1402	2000
610	B2 = FUN1CXW2CB,NW2CB,ZB2,WA2C];	3538	4148
122	WB25 = (B1+B2)*WA2;	4148	4270
30	WA3 = WA2-WB25;	4270	4300
598	WXQ2 = FUN1CXW2C,NW2C,ZWXQ2,WA2C];	3538	4136
130	WB3 = (WXQ2+S0025)*WA2;	4136	4266
974	WA31 = SQRT(SWA31*WS3*(P3-P41));	246	1220
142	H3 = SH3T3*T3-S84;	0	142
224	FAR41 = WF/WA31;	1220	1444
346	H41 = (H3+KDHB*FAR41)/(S1+FAR41);	1444	1790
122	T41 = S3Z90*H41+S308;	1790	1912
122	THTA41 = STH41*T41+S0856;	1912	2034
838	W41 = KWGTRB*(P41/SQRT(THTA41));	2034	2872
230	PR45Q1 = P45/P41;	0	230
598	DHQTH4 = FUN1CXPRGT,NPRGT,ZDHGTQ,PR45Q1];	230	828
118	DH41 = DHQTH4*THTA41;	2034	2152
390	TORQ41 = ((STRQ41*DH41)/NG)*W41;	2872	3262
134	T25Q2 = S115+S039*PS3Q2;	572	706
118	T25 = T25Q2*T2;	706	824
114	H25 = S240*T25;	824	938
126	H2 = S239*T2;	0	126
594	TORQC = STORQC*((H3*WA3-H2*WA2+H25*WB25)/NG);	4300	4894
54	H44 = H41-DH41;	2152	2206
114	H45 = S96Z3*H44;	2206	2320
122	T45 = S3537*H45+S1745;	2320	2442
230	PR49Q5 = P49/P45;	0	230
598	W45C = FUN1CXPRPT,NPRPT,ZW45C,PR49Q5];	230	828
122	THTA45 = STH45*T45+S0856;	2442	2564
852	W45 = (P45/SQRT(THTA45))*W45C;	2564	3416
610	DHQTH5 = FUN1CXPRPT,NPRPT,ZDHPTQ,PR49Q5];	230	840
118	DH45 = DHQTH5*THTA45;	2564	2682
390	TORQ45 = ((STRQ45*DH45)/NP)*W45;	3416	3806
42	H49 = H45-DH45;	2682	2724
578	NGDT = SNGDT*(TORQ41-TORQC);	4894	5472
560	NPDT = SNPDT*(TORQ45-TORQLD);	3806	4366
338	P41DT = (KV41*T41)*(WA31-W41+WF);	2872	3210
494	P45DT = (KV45*T45)*(W41-W45+S7826*WXQ2*WA2);	4136	4630
502	T3DT = ST3LG*(T3C-T3);	1302	1804
130	WS3DT = WA3-WB3-WA31;	4300	4430
204	NG = ADAMSENG,NGDT,DELTAT,SFTNG];	5472	5676
204	NP = ADAMSENP,NPDT,DELTAT,SFTNP];	4366	4570
196	P41 = ADAMSEP41,P41DT,DELTAT,SFTP41];	3210	3406
198	P45 = ADAMSEP45,P45DT,DELTAT,SFTP45];	4630	4828
184	T3 = ADAMSET3,T3DT,DELTAT,SFTT3];	1804	1988
206	WS3 = ADAMSEWS3,WS3DT,DELTAT,SFTWS3];	4430	4636

Figure 13. - "CAN START" and "CAN END" times (in compute cycles) for simulation equations.

```
*** PROCESSOR A ***;

*** ENGINE DYNAMICS ***;
P3 = KRQV3*WS3*T3;
PS3 = SPST3*P3;
PS3Q2 = PS3/P2;
H3 = SH3T3*T3-S84;
*** XFER H3 ***;
T25Q2 = S115+S039*PS3Q2;
T25 = T25Q2*T2;
H25 = S240*T25;
T3Q2 = FUN1(XPRC1,NPRC1,ZTRC,PS3Q2);
T3C = T3Q2*T2;
*** VERIFY PCNGC ***;
B1 = FUN1(XPNGC,NPNGC,ZB1,STSE4CH2.P,PCNGC);
*** UPDATE DERIVATIVE ***;
TERM7 = T3C-T3;
T3DT = ST3LG*TERM7;
*** UPDATE INTEGRATOR ***;
T3 = ADAMSC(T3,T3DT,DELTAT,SFTT3);
DEL2 = P2/PSTD;
*** WAIT 2 ***;
*** VERIFY WA2C ***;
*** VERIFY RTTH2 ***;
WA2 = (STSE4CH2.P.WA2C/STSE4CH2.P.RTTH2)*DEL2;
*** XFER WA2 ***;
*** REVERIFY WA2C ***;
WXQ2 = FUN1(XW2C,NW2C,ZWXQ2,STSE4CH2.P.WA2C);
*** XFER WXQ2 ***;
WB3 = (WXQ2+S0025)*WA2;
*** UPDATE DERIVATIVE ***;
*** VERIFY WA3 ***;
*** VERIFY WA31 ***;
WS3DT = STSE4CH2.P.WA3-WB3-STSE4CH3.C.WA31;
*** UPDATE INTEGRATOR ***;
WS3 = ADAMSC(WS3,WS3DT,DELTAT,SFTWS3);
```

Figure 14. - Quad processor distribution of simulation equations.

```
*** PROCESSOR B ***;

*** ENGINE DYNAMICS (CRITICAL PATH) ***;
RTTH2 = SQRT(T2/TSTD);
NGC = NG/RTTH2;
PCNGC = NGC/SNDES;
*** VERIFY PS3Q2 ***;
WA2C = MAP(XPRC,YPNGC,NPRC,ZW2C,STSE4CH2.C.PS3Q2,PCNGC);
B2 = FUN1(XW2CB,NW2CB,ZB2,WA2C);
*** VERIFY B1 ***;
*** VERIFY WA2 ***;
WB25 = (STSE4CH2.C.B1+B2)*STSE4CH2.C.WA2;
*** REVERIFY WA2 ***;
WA3 = STSE4CH2.C.WA2-WB25;
*** VERIFY H3 ***;
*** VERIFY H2 ***;
*** REVERIFY WA2 ***;
TERM2 = STSE4CH2.C.H3*WA3-STSE4CH3.P.H2*STSE4CH2.C.WA2;
*** VERIFY H25 ***;
TORQC = STORQC*((TERM2+STSE4CH2.C.H25*WB25)/NG);
*** UPDATE DERIVATIVE ***;
*** VERIFY TORQ41 ***;
TERM3 = STSE4CH3.P.TORQ41-TORQC;
NGDT = SNGDT*TERM3;
*** UPDATE INTEGRATOR ***;
NG = ADAMSC(NG,NGDT,DELTAT,SFTNG);
```

Figure 14. - Continued.

```
*** PROCESSOR C ***;

*** ENGINE DYNAMICS ***;
*** VERIFY WS3 ***;
*** VERIFY T3 ***;
P3 = KRQV3*STSE4CH2.C.WS3$1*STSE4CH2.C.T3$1;
*** VERIFY P41 ***;
TERM1 = P3-STSE4CH3.P.P41$1;
*** REVERIFY WS3 ***;
WA31 = SQRT(SWA31*STSE4CH2.C.WS3$1*TERM1);
*** XFER WA31 ***;
*** VERIFY WF ***;
FAR41 = STSE4CH1.P.WF/WA31;
*** VERIFY H3 ***;
H41 = (STSE4CH2.C.H3+KDHB*FAR41)/(S1+FAR41);
T41 = S3298*H41+S308;
THTA41 = STH41*T41+S0856;
*** VERIFY DHQTH4 ***;
DH41 = STSE4CH3.P.DHQTH4*THTA41;
H44 = H41-DH41;
H45 = S9623*H44;
T45 = S3537*H45+S1745;
THTA45 = STH45*T45+S0856;
*** VERIFY P45 ***;
*** VERIFY W45C ***;
W45 = (STSE4CH3.P.P45$1/SQRT(THTA45))*STSE4CH3.P.W45C;
*** VERIFY DHQTH5 ***;
DH45 = STSE4CH3.P.DHQTH5*THTA45;
TORQ45 = ((STRQ45*DH45)/NP)*W45;
H49 = H45-DH45;
*** UPDATE DERIVATIVE ***;
*** VERIFY TORQLD ***;
TERM4 = TORQ45-STSE4CH1.P.TORQLD;
NPDT = SNPDT*TERM4;
*** UPDATE INTEGRATOR ***;
NP = ADAMSEQNP,NPDT,DELTAT,SFTNPJ;
```

Figure 14. - Continued.

```
*** PROCESSOR D ***;

*** ENGINE DYNAMICS ***;
PR45Q1 = P45/P41;
DHQTH4 = FUN1CXPRGT,NPRGT,ZDHQTQ,PR45Q1J;
PR49Q5 = P49/P45;
W45C = FUN1CXPRPT,NPRPT,ZW45C,PR49Q5J;
DHQTH5 = FUN1CXPRPT,NPRPT,ZDHPTQ,PR49Q5J;
H2 = S239*T2;
*** XFER H2 ***;
*** WAIT 84 ***;
*** VERIFY THTA41 ***;
W41 = KWGTR8*(P41/SQRT(STSE4CH3.C.THTA41));
*** VERIFY DH41 ***;
*** VERIFY NG ***;
TORQ41 = ((STRQ41*STSE4CH3.C.DH41)/STSE4CH2.P.NG$1)*W41;
*** XFER TORQ41 ***;
*** UPDATE DERIVATIVE ***;
*** VERIFY WA31 ***;
TERM5 = STSE4CH3.C.WA31-W41;
*** VERIFY T41 ***;
P41DT = (KV41*STSE4CH3.C.T41)*(TERM5+STSE4CH1.P.WF);
*** UPDATE INTEGRATOR ***;
P41 = ADAMSEP41,P41DT,DELTAT,SFTP41J;
*** UPDATE DERIVATIVE ***;
*** VERIFY W45 ***;
TERM6 = W41-STSE4CH3.C.W45;
*** VERIFY T45 ***;
*** VERIFY WX02 ***;
*** VERIFY WA2 ***;
P45DT = (KV45*STSE4CH3.C.T45)*(TERM6+
                               S7826*STSE4CH2.C.WX02*STSE4CH2.C.WA2);
*** UPDATE INTEGRATOR ***;
P45 = ADAMSEP45,P45DT,DELTAT,SFTP45J;
```

Figure 14. - Concluded.

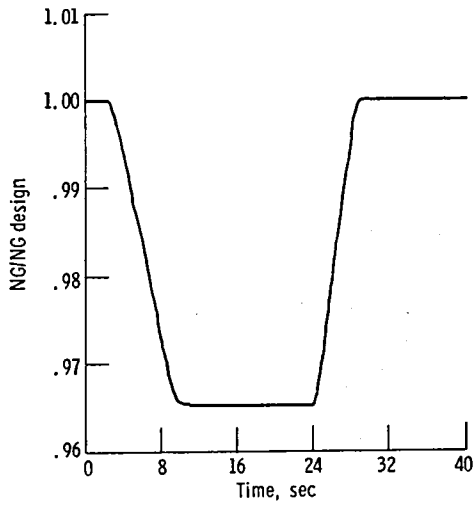


Figure 15. - Gas generator turbine speed versus time. Quad processor.

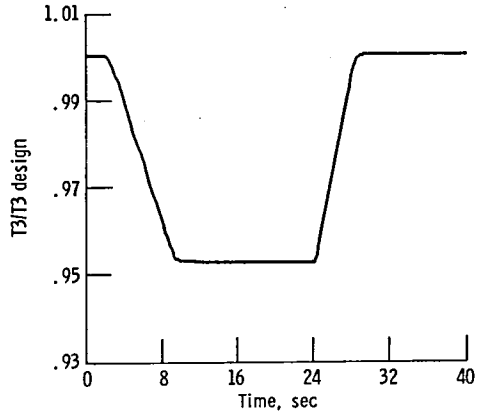


Figure 16. - Compressor discharge temperature versus time. Quad processor.

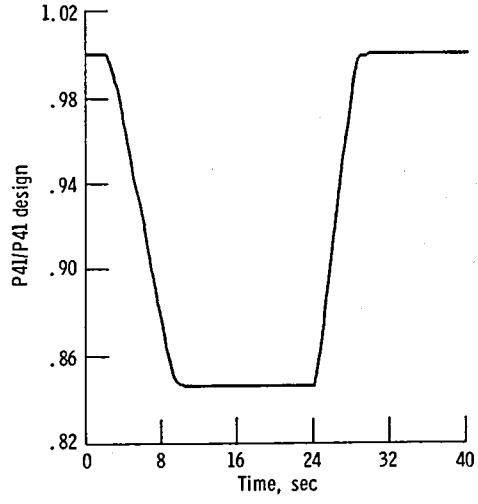


Figure 17. - Gas generator turbine inlet pressure versus time. Quad processor.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-87216		

4. Title and Subtitle	5. Report Date
Simulating a Small Turboshaft Engine in a Real-Time Multiprocessor Simulator (RTMPS) Environment	6. Performing Organization Code: 505-40-74

7. Author(s)	8. Performing Organization Report No.
Edward J. Milner and Dale J. Arpasi	E-2876
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135	13. Type of Report and Period Covered: Technical Memorandum

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546	

15. Supplementary Notes

Material not previously published, presented at the 1986 SCS Multiconference, sponsored by the Society for Computer Simulation, San Diego, California, January 23-25, 1986.

16. Abstract

A Real-Time Multiprocessor Simulator (RTMPS) has been developed at NASA Lewis Research Center. The RTMPS uses parallel microprocessors to achieve computing speeds needed for real-time engine simulation. This report describes the use of the RTMPS system to simulate a small turboshaft engine. The process of programming the engine equations and distributing them over one, two, and four processors is discussed. Steady-state and transient results from the RTMPS simulation are compared with results from a main-frame-based simulation. Processor execution times and the associated execution time savings for the two and four processor cases are presented using actual data obtained from the RTMPS system. Included is a discussion of why the minimum achievable calculation time for the turboshaft engine model was attained using four processors. Finally, future enhancements to the RTMPS system are discussed including the development of a generalized partitioning algorithm to automatically distribute the system equations among the processors in optimum fashion.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Digital Simulator; Parallel processing; Real-time Simulator	Unclassified - unlimited STAR Category 07

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified		

*For sale by the National Technical Information Service, Springfield, Virginia 22161

End of Document