SIMULATION OF THE ILLIAC IV - B6500 REAL-TIME COMPUTING SYSTEM

H. Robert Downs
Systems Control, Inc.

Norman R. Nielsen
Stanford University

Edward T. Watanabe
Systems Control, Inc.

Abstract

The ILLIAC IV, a fast parallel-array computer, is being considered as a control computer for a large radar system. A high-level simulation of the ILLIAC is being done to determine the ILLIAC's effectiveness in such a system and to investigate various design alternatives.

The work to be performed by the system is divided into "tasks." Timing constraints for the tasks are input to the simulator along with a list of tasks to be executed in a certain period of time.

A number of resources (e.g., memory, program, data) must be assembled prior to the execution of each task. The model can then be used to test the effectiveness of various scheduling strategies under a variety of loads and the results will be used to evaluate the overall effectiveness of ILLIAC IV for this system.

I INTRODUCTION

The ILLIAC IV computer, a highly parallel array processor, represents an architectural concept which is quite distinct from conventional computer architectures. The intent of this design is to provide computing throughput which is orders of magnitude above that which is currently available. This drastic departure from familiar concepts requires different means for determining the applicability of this design to specific user requirements.

The purpose of this simulation was to determine the applicability of the ILLIAC IV for controlling a large and complex real-time system. This system is very large and requires quite a bit of data processing power. In fact, no present generation machine is capable of performing the required functions for this system in the necessary time frame. The main goal of this simulation project was to determine whether this real-time system could be satisfactorily controlled by an ILLIAC IV computer system.

Since the ILLIAC IV is a highly-specialized parallel processor, it is very inefficient at certain tasks. Unfortunately, the tasks which constitute an operating system fall largely into this class. Therefore, in order to construct an efficient configuration, the ILLIAC IV is controlled by a general purpose control processor - in this case a Burroughs B6500. The B6500 contains the major portion of the operating system and performs most of the processing associated with it. The B6500 also performs a number of other tasks for the real-time system; these tasks are those which can be handled most effectively on a computer having contemporary architecture. The simulator was also used to investigate the adequacy of the B6500 for these various control and processing functions.

The simulation also showed how the control of the ILLIAC IV could be accomplished. That is, it was used to investigate the sufficiency of the proposed operating system and I/O configuration as well as the B6500 control computer(s).

II DESCRIPTION OF THE SYSTEM

A. Actual System

1. ILLIAC IV - The ILLIAC IV computer is a parallel array of 256 coupled processing elements (PE's) arranged in four quadrants. This study, however, addressed a one quadrant configuration. (This configuration is currently being constructed by the Burroughs Corporation for the University of Illinois.) The quadrant contains 64 PE's which are driven by decoded instruction signals emanating from a single control unit (CU) working on a single instruction stream.

The PE's of a quadrant must simultaneously carry out the same operation on the operands to which they each have access. Thus the PE's will operate in parallel. Each of the 64 PE's will have an instruction set which includes floating point arithmetic on both 64-bit and 32-bit operands (with options for rounding and normalization), 8-bit operations, and a wide range of tests due to the use of addressable registers and a full set of comparisons. The expected time for a 64-bit floating point add execution is 250 nanoseconds and for a 32-bit add is 300 nanoseconds. The PE's differ from conventional computers in three main ways. Firstly, each is capable of communicating data to its four neighboring PE's in the array by means of routing instructions. Secondly, each PE is able to set its own mode registers thus enabling or disabling itself for the transmission of data or the execution of instructions from its CU. Thirdly, all instruction decoding and some other functions are effected in the CU, thus eliminating the need for a large amount of hardware in the PE's. Each CU contain a 64-word instruction buffer, a 64-word local data buffer and 4 accumulator registers which are loaded, as required, from memory.[1]

2. Computing Configuration - In addition to the ILLIAC IV, the system being simulated consists of a Burroughs B6500 as a control computer, a Burroughs I/O Control Unit (IOC), a one billion bit disc storage unit (SU) with associated Electronic Unit (EU), and a Buffer I/O Memory (BIOM) of 8192 words to facilitate B6500 - ILLIAC memory to memory data transfers in the light of the disparate sizes of the memory access paths. There is an I/O Switch in the IOC which handles data transfers between the special ILLIAC IV I/O devices and the PE memories at a rate of 1024 bits per microsecond and which co-ordinates all other I/O to and from the PE memories.

This entire configuration is used to control a large real-time system which is interfaced with the computing system through a Real-Time Interface Processor (RIP). This processor buffers the data flow into and out of the ILLIAC IV via the IOC and also performs some special purpose data processing.

3. The Real-Time System - The real-time system being controlled by the ILLIAC IV in this simulation is a large electronically steered radar. The ILLIAC is required to perform large data processing tasks in a few milliseconds, to schedule and format orders to the radar and to respond in a few milliseconds to large and unpredictable amounts of data returned from the radar. The real-time interface processor performs a variety of special purpose functions such as digital-to-analog and analog-to-digital conversion as well as buffering of data sent from the ILLIAC IV to the RIP or vice versa. In effect, the tasks to be performed by this system usually consist of (1) the ILLIAC performing some pre-processing; (2) a block of orders being sent to and subsequently executed by the RIP; (3) response data being forwarded from the RIP to the ILLIAC; and (4) some post-processing of the new data by the ILLIAC.

A diagram of this system is shown in Figure 1.

B. Model of the System:

The system which was simulated consisted of the ILLIAC IV itself, the B6500 control computer, the buffer I/O memory (BIOM), the I/O controller (IOC), and the real-time interface processor (RIP). Each of these devices is modeled in terms of its memory size, its I/O rate, and the tasks which must be executed upon it. The simulation was written in a modular fashion so that each of these devices was simulated separately. This permits the operations or even the functions of a device to be changed without adjustment of the logic in other portions of the simulator.

Because of the virtual memory concept of the B6500 system and because of the undefined nature of its non-operating system processing tasks, the constraints of memory space and I/O load were ignored. Thus, the B6500 was characterized only in terms of available processing time. One of the parameters for the system specified the number of central processors (up to four) which could be used for the control computer. The B6500 executes all of the non-parallel operating tasks for the ILLIAC IV. Thus, the B6500 controls all communication with the ILLIAC, including all I/O transfers of programs and data. In addition it issues all scheduling and control commands to the ILLIAC. Brief commands are sent through a special 48-bit control line while longer data transfers take place via the BIOM.

C. Model of Processing Load

The processing load was modeled from a specification of the following types of imputs to the simulator:

(1) Tasks - Each task which the ILLIAC IV must execute is described by the parameters of computing time, memory space requirements, data transfers required before execution, and any prerequisite tasks. In addition, a number of post-execution operations may be specified, such as the transfer of data from the ILLIAC to the RIP.

(2) Working Spaces - The working spaces reflect the various sections of ILLIAC memory which are required by the tasks in order to execute. These include space for data sets, buffer areas, and scratch areas as well as for the program itself.

(3) Data Transmissions - Various tasks require that certain data transmissions take place either before or after their execution on the ILLIAC IV. These transmissions are defined in terms of their size, origination, destination, triggered replies, etc.

(4) RIP Tasks - These tasks are described by the amount of RIP processing required, the time at which it is required, and the data which will result from the object system as a result of that processing.

The simulation was designed to determine the effectiveness of the ILLIAC IV computing system for controlling the real-time system without the necessity for coding and executing either the operating system or the computing tasks for the ILLIAC and the RIP. However, it was still necessary to develop estimates of the processing, I/O, and memory requirements of the various ILLIAC tasks. Data was also needed on the RIP and B6500 processing requirements. Some of the more frequently executed tasks were investigated in detail and flow charts describing them were constructed. Sample code was generated for portions of these tasks in order to determine estimates of the computing time and file usage that might be expected for a parallel-array computer. These investigations were made simultaneously with the simulation development effort. It must be recognized that these inputs are fairly gross estimates and may contain a bias toward high or low memory and/or computing time estimates. To give credence to the simulation results, the sensitivity of the system's performance as a function of these estimates is being studied.

III EXECUTION OF THE SIMULATOR

A. Effectiveness Measures

Under the proposed mode of operation the ILLIAC IV does not respond to a real-time interrupts (as do many real time systems). Rather, it must dynamically assess the future processing load and allocate processing time accordingly. The B6500 then tries to implement this schedule, calling upon the ILLIAC IV as necessary to provide the requisite processing. It should be noted that the effectiveness of the future load assessment procedure was not studied by this simulation. Rather, the ability of the system to handle a load once specified was evaluated.

The effectiveness of the ILLIAC IV was assessed in the following manner: (1) a particularly heavy real-time processing load was hypothesized; (2) the corresponding data processing tasks were then determined, and a crude schedule indicating their order of execution was constructed (i.e., the ILLIAC load assessment); and (3) this schedule list was input to the simulator, and the modeled system attèmpted to implement it. The degree to which the system could remain "on schedule" was taken as the measure of effectiveness.

B. Operation of the Simulator

Because of the volume of data inputs, a special data description language was developed. The first portion of the simulator takes these "English-like"inputs, checks them for errors, and initializes the simulator. During the course of the execution of the model, a number of statistics are calculated and printed covering the performance of the simulated system. In addition, a trace file of every major change of state in the model can be obtained. Not only does this permit post-mortem examination of system performance, but it serves as the input for a number of plotting routines.

The evaluation of the simulation results was greatly enhanced by two sets of time history plots. The first set of plots shows the execution of tasks (transmissions in case of RIP), over time in each of the modeled devices. The other set of plots shows various activity indicators., for each device, such as, (1) memory utilization, and active/idle status for the ILLIAC IV; (2) job queue status and active/idle status for the RIP; (3) CPU status (for each fourteen polling function) in the B6500; (4) disc file controller status and priority queue status in the IOC; and (5) memory utilization and request queue status in the BIOM.

One of the requirements for the simulation model was that it be operable on several different computers. Despite both the appropriateness and the desirability of using a higher level simulation language for the model, there was no one such language that was available on all of these computers. Thus, the model was constructed in FORTRAN IV (which persists as the "universal" language). This language choice made model development much more of a chore, but it did enable the model to be run on different computers without reprogramming. Presently the model is implemented on the IMB 360, the UNIVAC 1108, and partially on the GE 635.

For the purpose of indicating resource requirements, the following figures are based on the use of UNIVAC 1108 and the UNIVAC FORTRAN V compiler. The largest portion of the program is the data input and initialization section. This routine requires approximately 22K words (36 bit), but is easily overlayable, being used but once per run. The simulation itself requires only 11K words with an additional 11k words being required for the FORTRAN I/O routines. The storage requirements for the simulator's tables, pointers, and queues require a further 18K words. In total, then, a partition size of 51K words is required, although this could very easily be reduced to half that amount without significantly affecting execution time.

With the job descriptions currently being employed, the simulator executed 170 times slower than the real system when a full trace was being made. However, when tracing is disabled the simulator runs only 1.4 times slower than the real system. (The execution of the simulator is not I/O bound, even with tracing, this comparison provides an interesting view of the processing efficiency of the FORTRAN I/O package). Considering that the simulator actually represents four different simulation models and considering that some of the simulated devices operate considerably faster than the UNIVAC 1108, the optimized 1.4 execution factor is quite encouraging.

The collection of input data consumed about three man-months of effort. The development and exercise of the model required an additional twelve man-months of effort.

C. Experiments Performed with the Simulator

In order to answer the primary question addressed by this simulation, "Can the ILLIAC IV control this system?", the effects of all parts of the configuration other than the ILLIAC IV and the RIP were minimized. This was achieved by setting all of the processing and overhead times required by the remaining devices to one microsecond (an insignificant amount of time for those portions of the system). This action permitted a determination of (1) whether there was sufficient computation power in the ILLIAC IV to meet the processing requirements and (2) whether or not the real-time constraints imposed by the system were achievable with possible I/O control configurations.

The plots in Figures 2 and 3 are indicative of the data resulting from this experiment. Together these two figures indicate that the ILLIAC IV system would be able to meet the real-time operating constraints when using a carefully developed schedule. The RIP subsystem reached the desired 100% level of utilization, as can be seen from the active/idle status line in Figure 3. Further, this was achieved with only a 50% utilization of the ILLIAC, as can be seen in Figure 2. Thus, even for a particularly heavy load situation, the ILLIAC IV appears to have ample capacity, so long as data transmission and operating systems overheads do not needlessly delay the execution phases.

On the other hand, the main ILLIAC IV memory (2048 words per processing element) does appear somewhat limiting. While the memory utilization ranges from 60% to 85% in the particular simulations described, it is quite possible that a heavier peak load (with high memory requirements) could occur. This is not unrealistic when one considers that the sizes of some of the individual working spaces for a task may exceed 10% of the total PE memory space. Further, as was mentioned above, the memory estimates for each of the tasks may be potentially somewhat low. Various alternatives are being considered, such as expanding or paging the memory, in order to alleviate this situation.

It should also be noted that the "wait" status lines in the plots in Figures 2 and 3 refer to the lengths of the queues of job awaiting processing on the ILLIAC IV and the RIP. Inasmuch as the jobs have early start times associated with them, it is not inconsistent for a device to be idle while simultaneously facing a non-empty queue.

A further simulation was made with the processing and overhead times of the remaining devices in the configuration restored to their estimated correct value.

Again, the ILLIAC IV system appeared able to handle the processing workload within the desired time constraints. However, it was necessary to use a dual processor B6500 in order to meet the desired schedule. Figure 4 indicates the processing activity which took place on the control processor. In addition to the activity lines for each of the 11 control functions, the utilizations of the two CPU's are also plotted. From this data it is clear that the problem is not so much one of insufficient computer power as it is one of bunched computing requirements. Given an appropriate distribution of processing demands, a single processor would nearly suffice. Attention is currently being devoted to the problem of eliminating or at least minimizing the bunching of requests.

A second question which has been addressed by the simulator is, "What type of configuration is sufficient to allow the system to perform effectively?" A number of possible configurations have been suggested. Some of these were simulated by slight modifications of the current simulator. However, some did not require any modification. For example, the use of a multiprocessor B6500 system could be tested by appropriate parameter adjustment. Another type of configuration change which is readily permitted is a change in the type and capacity of the various I/O devices. For instance, the disc secondary storage could be replaced by a faster access bulk storage device. This would be useful in those cases where working spaces must frequently be swapped into and out of ILLIAC IV memory. Aslo, the bandwidth of certain data busses between devices may be varied to obtain a more effective system performance.

Finally, certain changes to the operating system rules may be easily effected in the simulator.

IV CONCLUSIONS

From the results of this simulation we conclude that the ILLIAC IV does have sufficient computational power to control the real-time system in question. However, it would appear that the expansion of ILLIAC IV memory from 2048 to 4096 words per processing element would be appropriate in order to assure that serious time delays resulting from insufficient memory space would be minimized. It also appears that a dual processor B6500 system will be adequate for control of the ILLIAC IV system.

The simulation will continue to be exercised in order to answer several more subtle questions. These include investigations of possible alternatives to (1) the current operating system philosophy, (2) the B6500 as a control computer, and (3) the use of secondary storage to alleviate memory problems in the ILLIAC IV.

This simulation has shown itself to be a worthwhile investment. It has already provided quantitative answers to some very difficult questions. In the future it will serve as a valuable design tool when more specific design questions are addressed.

REFERENCES

1. ILLIAC IV Systems Characteristics and Programming Manual. Burroughs Corporation, Defense Space and Special Systems Group, IL4-PM1, 30 January 1970.

BIOGRAPHIES

1. H. Robert Downs is a Senior Research Engineer with Systems Control, Inc. He is currently involved in studying the ILLIAC IV computer and its possible application to a real-time problem. Mr. Downs received his B.A. and M.A. from the University of California at Los Angeles and his Ph.D from the University of California, Berkely.

2. Norman R. Nielsen is an Associate Professor of Operations and Systems Analysis in the Stanford Graduate School of Business and is also Associate Director for Planning of the Stanford Computation Center. His previous work in the area of computer system simulation includes a study of the IBM 360/67 and the Burroughs B6500 systems. More recently he was the designer of ECSS, a language specifically for use in modeling computer systems. Mr. Nielsen recieved his B.A. from Pomona College, and his M.B.A. and Ph.D from Stanford University.

3. Edward T. Watanabe is a Research Mathematican with the Computer Systems Division of Systems Control, Inc. He has been involved in wide areas of computer application for the past 8 years. Mr. Watanabe received his B.A. from San Jose State College.

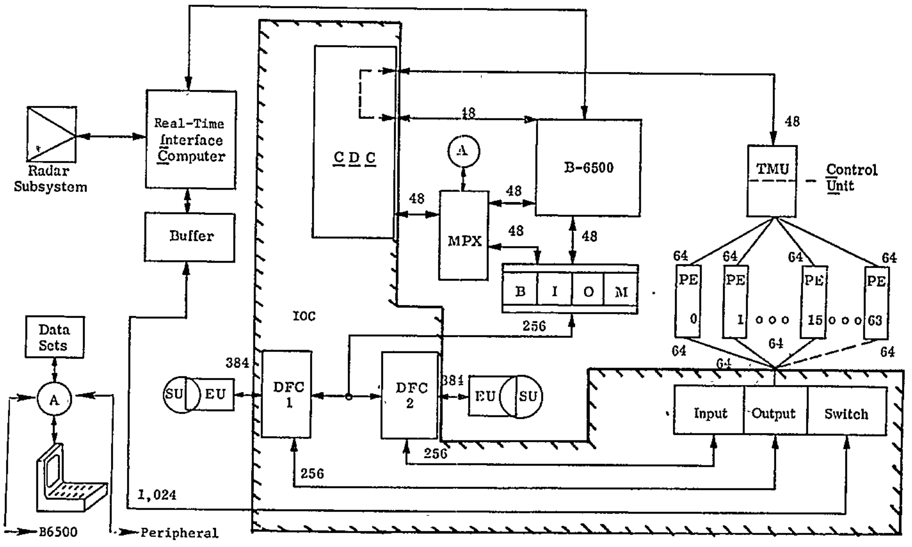
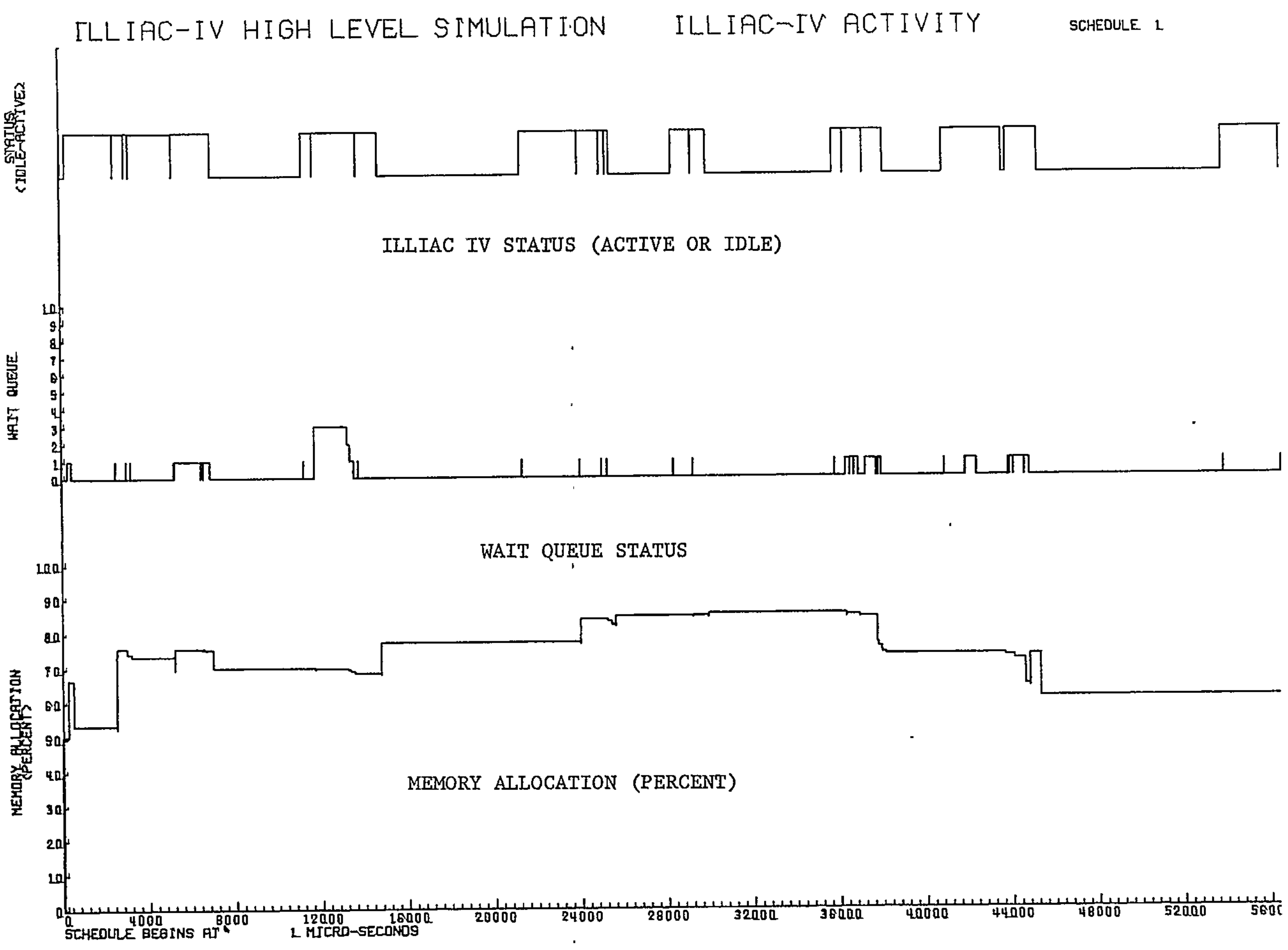
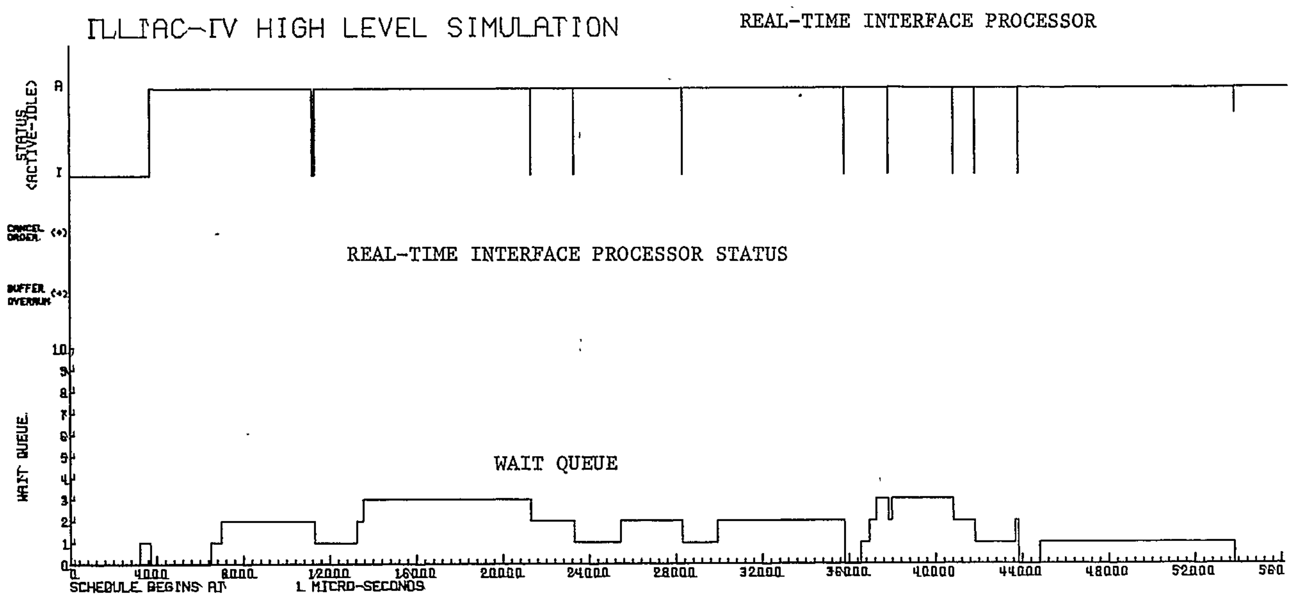


Fig. 1 AN ILLIAC-IV B6500 REAL-TIME CONFIGURATION

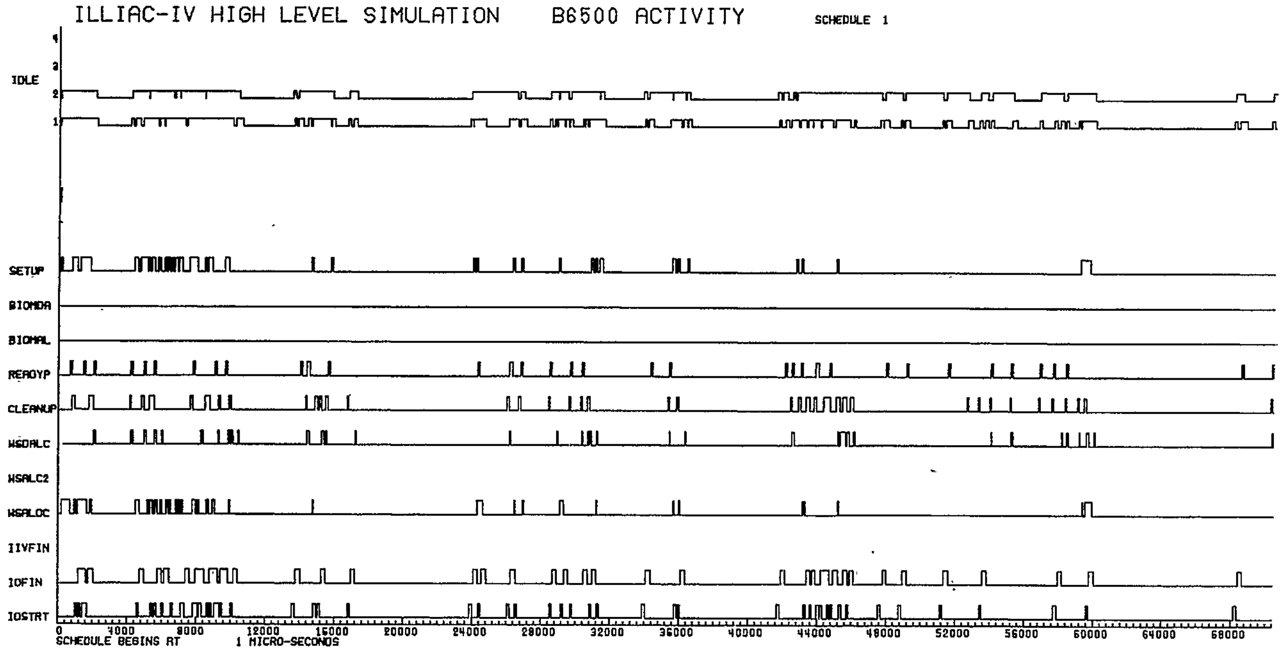


ILLIAC IV ACTIVITY PLOT (FIGURE 2)



REAL-TIME INTERFACE PROCESSOR ACTIVITY PLOT (FIGURE 3)

Task execution schedule was manupilated until approximately 100% utilization of RIP after initial start-up period was achieved.



B6500 ACTIVITY PLOT (FIGURE 4)

Activity profile of the B6500 computer with two processors and the 11 polling functions of an operating system is shown above.

Polling functions WSALC2 and IIVFIN were never called during the run, BIOMDA and BIONAL were called after the shown time period.

THE DEFINITION OF POLLING FUNCTIONS

IOSTRT - initiates all input-output operations.
IOFIN - handles the I/O completion interrupts from the IOC.
IIVFIN - handles the completion of move operations on the ILLIAC IV.
WSALOC - is the primary working space allocation function.
WSALC2 - is the secondary working space allocation routine.
WSDALC - is the responsible for the deallocation of working space.
CLENUP - processes the postcondition requirements of a task.
READYP - marks all of a task's dependent tasks as ready.
BIOMAL - allocates BIOM space for I/O requests between the ILLIAC and the B6500.
BIOMDA - deallocates BIOM space.
SETUP - processes a task's preconditions.