1981 Winter Simulation Conference Proceedings
T.I. Ören, C.M. Delfosse, C.M. Shub (Eds.)

A COMPARISON OF METHODS FOR SIMULATING COMPUTER BUS ARCHITECTURES

Larry Wear, Ph.D.
Professor of Computer Science
Ron Guilmette, Mike Spann, Mike Chiusolo
Department of Computer Science
California State University, Chico
Chico, California 95929

This paper describes three methods that were used to investigate multiprocessor bus architectures. The models described were implemented in FORTRAN, GPSS, and SIMULA. Characteristics of the three implementations, such as program length, program memory requirements, execution time and ease of use are compared. Results of the simulation of a single bus system are presented to show how the various parameters affect system performance.

1. INTRODUCTION

The recognition of the importance of simulation led to the development of several languages and support packages as the tools for simulation, e.g. CSL, DMOS, DEMOS, GASP, GPSS, GSP, SIMSCRIPT, SIMPAC and SIMULA. Each of these has its own characteristics, advantages, and disadvantages.

This paper documents the simulation of four computer architectural models in three different languages. The models simulate the impact of different types of bus architectures on a multiprocessor system. The four bus architecture models that were simulated are the following: UNIBUS, DUAL BUS, MULTIBUS and CROSS BAR SWITCH (Leibowitz 1978). The simulations of each of these bus architecture models were written in three different languages--FORTRAN, GPSS and SIMULA.

These models allow for a variable number of processors, memory modules, I/O units and buses. The cycle time of the processor, memory modules, the transfer rate of the bus, and the mean rate of I/O requests can be defined independently. The interarrival time of I/O requests may be based upon any arbitrarily selected distribution. Processors in the models execute three kinds of instructions which last for one, two, or three CPU cycles. The distribution of these instructions is also a user defined parameter.

The output from the simulation is the percentage of ACTIVE, BLOCKED, FROZEN and FREE time on each individual piece of hardware in the model. The statistics have been analyzed and transformed to graphical representations for easy interpretation.

2. FORTRAN SIMULATION

2.1 Description of FORTRAN Program

The FORTRAN simulation program is table driven. All parameters used by the program are contained in the state table, except for a few pieces of global data (i.e. clock, clock increment). One of the most powerful features of this program is the way in which the bus structure is defined in the state table by means of a bit map. This method allows virtually any type of architecture to be defined. The FORTRAN program is broken up into the following three main sections:

State table. Each item in the bus structure (this includes busses, memories, cpus and I/O units) is represented in a state table. Each unit has one entry in the table. This entry includes specific information pertinent to its operation (i.e. speed, interarrival time, write/read capability, priority, interleaving...etc.).

Clock. Before each clock increment, the state table is to be scanned. The clock is then incremented to the time of the next available unit in the table.

Statistics gathering. After each increment of the clock, the state table is scanned and each unit's entry is updated with how long it was ACTIVE, BLOCKED, FROZEN or FREE. Request queue lengths are also recorded.

2.2 Advantages of the FORTRAN Implementation

1. Everyone working on the project programmed in FORTRAN. This eliminated the need to learn a new language before developing the simulation.

81CH1709-5/81/0000-0353$00.75

2. The FORTRAN used was a subset of FORTRAN IV and therefore easily transportable to many computers, including most minis.

3. Since the simulation could be run on a local minicomputer, a HP1000F, turnaround time was very good.

4. Execution time was less than for the SIMULA and GPSS simulations.

2.2 Disadvantages of the FORTRAN Implementation

1. Because FORTRAN was not designed to be a simulation language, the program was quite long, approximately 500 lines.

2. Initial design was more complex because of the need to create many simulation primitives (i.e. clock, state changes...etc.).

3. Debugging the program was difficult. This was partially because the primitives that are provided in a simulation language had to be debugged in addition to the model itself.

2.3 Summary of the FORTRAN Program and Model

Because of the large size of this FORTRAN program, it was decided to incorporate the capability of simulating all of the different types of bus structures. This required much more variability and forethought into the initial design, and increased the debug and test phase of the program development.

3. GPSS SIMULATION

3.1 Description of GPSS Program

The GPSS model is currently capable of simulating the UNIBUS, MULTIBUS, and CROSS BAR switching system topologies. With minor modifications, it could simulate a multibus/multiported memory system.

The differences between the three currently implemented models are very minor. For example, the only difference between the MULTIBUS and the UNIBUS versions is the substitution of one line of code which defines the (constant) number of busses to be used during simulation. This value is simply set to one for the UNIBUS version. The modifications required to convert the MULTIBUS version to the CROSS BAR version consists of the substitution of two lines of code to provide for the fixed time overhead of cross-bar bus connection as opposed to the variable overhead required for the MULTIBUS system.

3.2 Advantages of GPSS Implementation

1. Because GPSS has many simulation primitives built into the language, (Schriber 1974), the program was short, about 150 lines of code.

2. Generation of output statistics was simplified because many GPSS statements cause statistics to be gathered automatically.

3. A variety of distributions can be specified for independent variables.

4. Because GPSS is interpretive, consistancy checks are automatically performed and this reduces development and debugging time.

3.3 Disadvantages of GPSS Implementation

1. Execution time for the model was relatively easy.

2. GPSS is not available on as many machines as FORTRAN and therefore the program is not as transportable as the FORTRAN version.

3.4 Summary of the GPSS Program and Model

The primitives available in GPSS made development of the program relatively easy. The features of the language made the program short and gathering statistics was also quite easy. The major disadvantage of the GPSS model was the long execution time.

4. SIMULA SIMULATION

4.1 Description of SIMULA Program

The SIMULA model was implemented using the DEMOS (Discrete Event Modeling on Simula) package developed by Dr. Graham Birtwistle (1981). University of Calgary using the activity diagram method he advocates. Using this approach, the model took 4-5 man hours to design, code and test.

Running approximately 150 lines of highly legible SIMULA code, it will run interactively, with the user defining the number of busses, CPU's, I/O modules, memory modules, and the timing characteristics of each module, or it can be run from a batch stream. The output is in the form of tables of numbers, histograms, and statistical analysis of the tables of numbers.

4.2 Advantages of SIMULA Implementation

1. SIMULA, a superset of ALGOL-60 is easy to code and very readable. (Birtwistle 1973).

2. The data collecting and reporting tools of DEMOS allows gathering of any data easily.

3. The tracing and debugging features of DEMOS simplify debugging modifications to the model, and varifying proper operation of the model.

4.2 Disadvantages of SIMULA Implementation

1. Being a high level, general purpose simulation language, models constructed using SIMULA are usually bigger (object size) and require more CPU time than a custom model in a lower level language.

2. Our SIMULA compiler generates extremely inefficient code, and some in some areas, of questionable reliability. This problem could be traced to the age of our compiler (it was written in 1968). A new compiler should be available soon so this should not be seen as a permanent problem.

3. SIMULA is available only on large machines

(CDC, IBM, DEC, VAC). As a result, the high cost of computer time for these large machines must be considered in program development.

4.3 Summary of SIMULA Model and Program

SIMULA offers an interesting (but often frustrating) range of trade-offs. On one hand it produces elegant, easily read and modified code. Modifications took an average of five to ten minutes for this program. On the other hand the inefficiency of the generated programs resulted in a constant worry about time limits, memory size and other such problems, rather than the correctness of the model.

5. RESULTS OF SIMULATIONS

The results for the three simulations were compared for the model shown in Fig. 1. The output from the three simulations were compared to verify correctness. All three simulations gave results that were within three percent for all parameters measured.

The results of the simulations are shown as plots of relative utilization of each of the functional units. The statistics gathered are FREE, FROZEN, BLOCKED and ACTIVE. Fig. 2-5 should be examined to see exactly how the states are defined for each module in the system. After some deliberation, we decided that by observing what percentage of the time each functional unit was in each of these states we could determine the relative efficiency of a given architecture with a minimum of numbers.

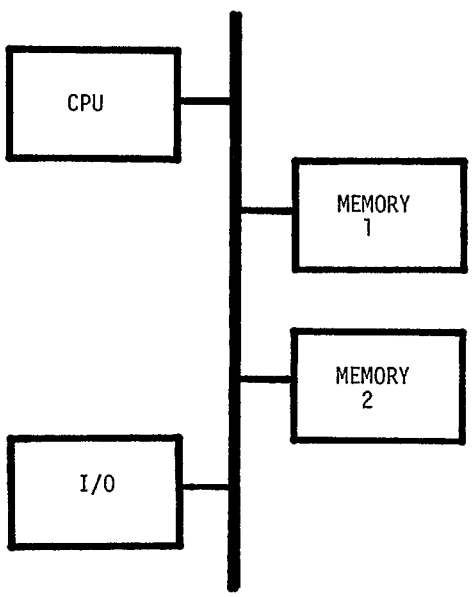

Fig. 1. Model for a Single Bus Two Memory Module System

Fig. 6 shows the percent ACTIVE time for the CPU, I/O, MEMORY and BUS as a function of I/O request frequency (number of instructions between requests). The figure shows that the CPU ACTIVE time drops to zero as the frequency of I/O requests increases.

The percent ACTIVE, as a function of memory cycle time, is shown in Fig. 7. As one would expect, when very fast memory is used the CPU can execute nearly 100% of the time. Also memory ACTIVE increases directly with memory cycle time.

CPU utilization is shown as a function of both memory cycle time and I/O request rate in Fig. 8. This figure confirms that CPU utilization increases as memory cycle time decreases and as I/O requests decrease.

6. CONCLUSIONS

Developing three separate models led to some interesting observations. The three simulations were developed on three different computer systems. The SIMULA and GPSS programs were written on CDC systems and the FORTRAN version was written for an HP system. Because of this it was not possible to make direct comparisons of the execution time for the different simulations. However, the FORTRAN version seemed to run several times faster than either the SIMULA or GPSS versions. This could be very significant if the simulations are going to be executed many times.

Both the SIMULA and GPSS simulations were much shorter than the FORTRAN version, by approximately four-to-one. There was a direct relation between the program length and the time required to develop the program. Besides being shorter than the FORTRAN program, the other two were easier to read and understand.

The three different programs were developed independently; this led to some interesting, if not surprising discoveries. The three teams developing the programs supposedly started with the same models. However, as soon as the first results were compared, it became obvious that there were significant differences in the results. This led us to take a more careful look at the assumptions upon which the models were based and we found that what one person thought was an insignificant modification of the model could cause definite differences in the results. If we had only developed one simulation it is quite possible that some of our results would have been in error. These errors would have been difficult to find since many of the results would have been nearly correct. Each simulation provided erroneous results along with valid ones. By attempting to resolve the areas of disagreement, some of the problems of the models were exposed. Activity diagrams described by Birtwistle were quite useful in finding subtle differences in the programs.

Developing simulations in both a simulation language and a general purpose language has several advantages. The SIMULA/GPSS versions can be implemented quickly and can serve as a design for the FORTRAN version. The FORTRAN version can be used when execution time and transportability are important. Finally, having two separate models gives

a good validity check on the programs.

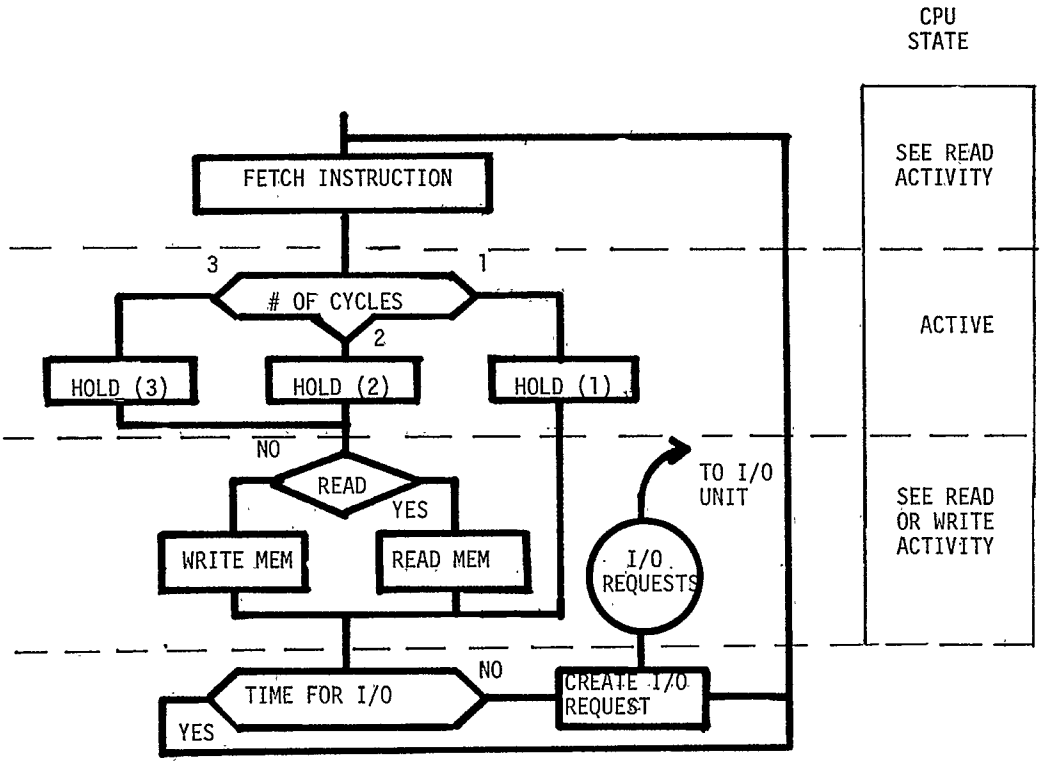

Fig. 2. CPU Activity Diagram

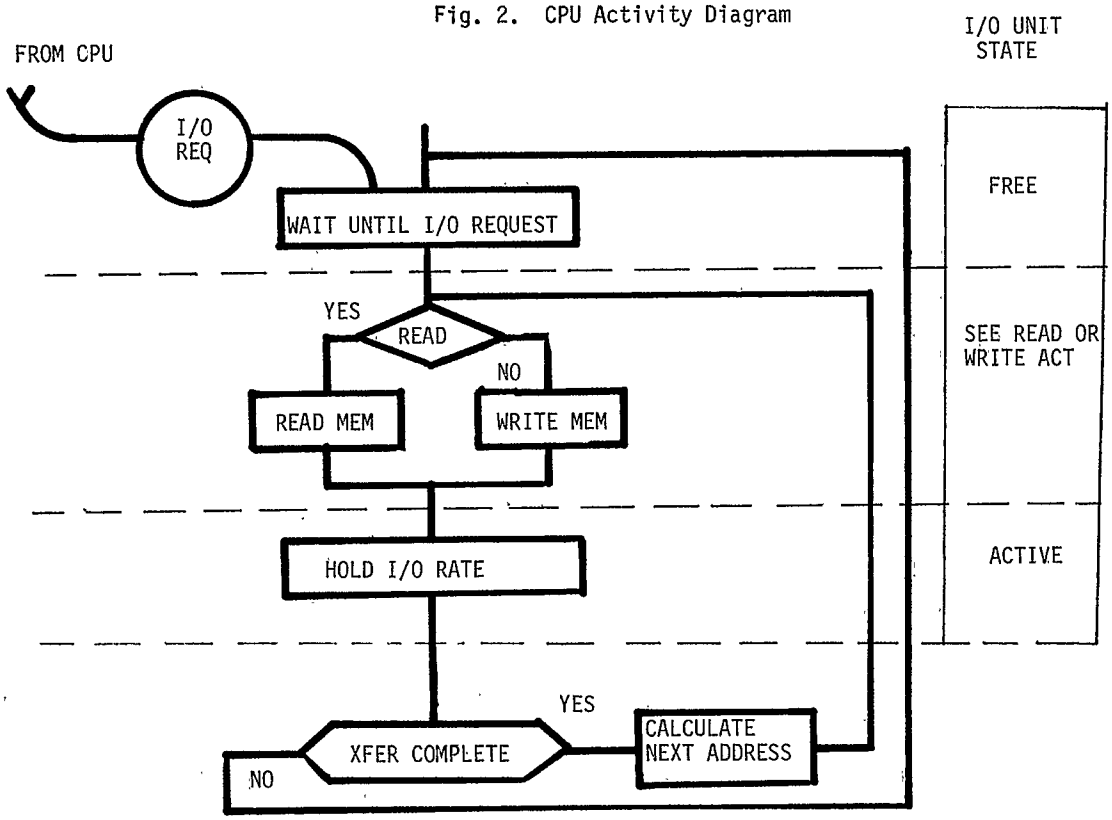

Fig. 3. I/O Unit Activity Diagram

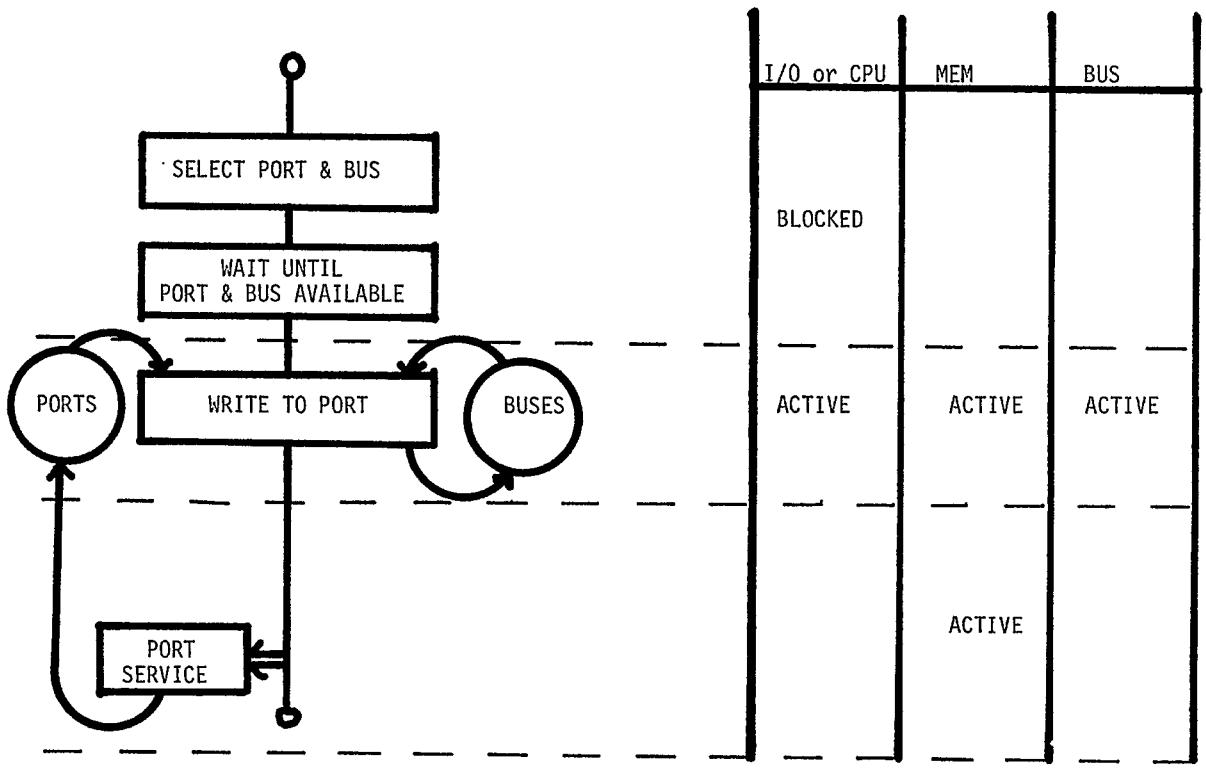

Fig 4. Write Activity Diagram

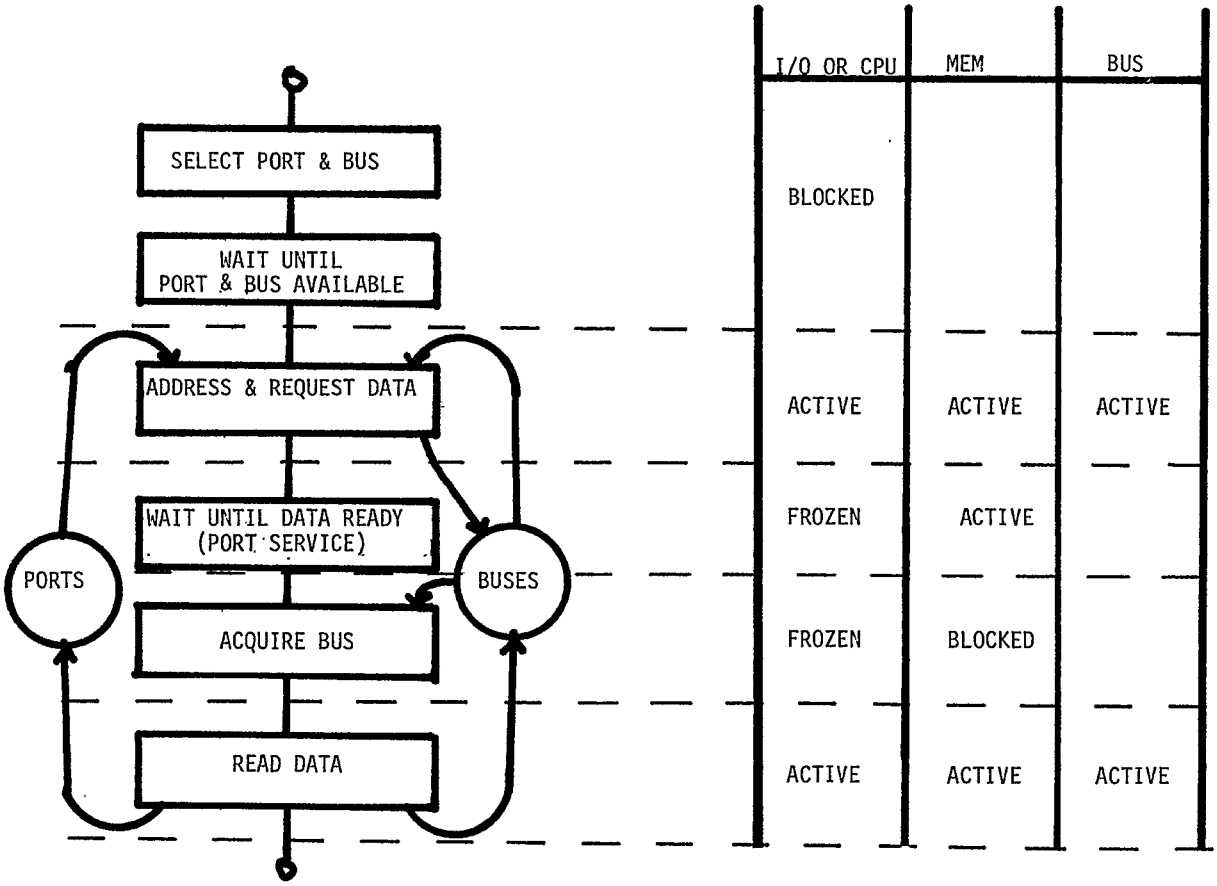

Fig. 5 Read Activity Diagram

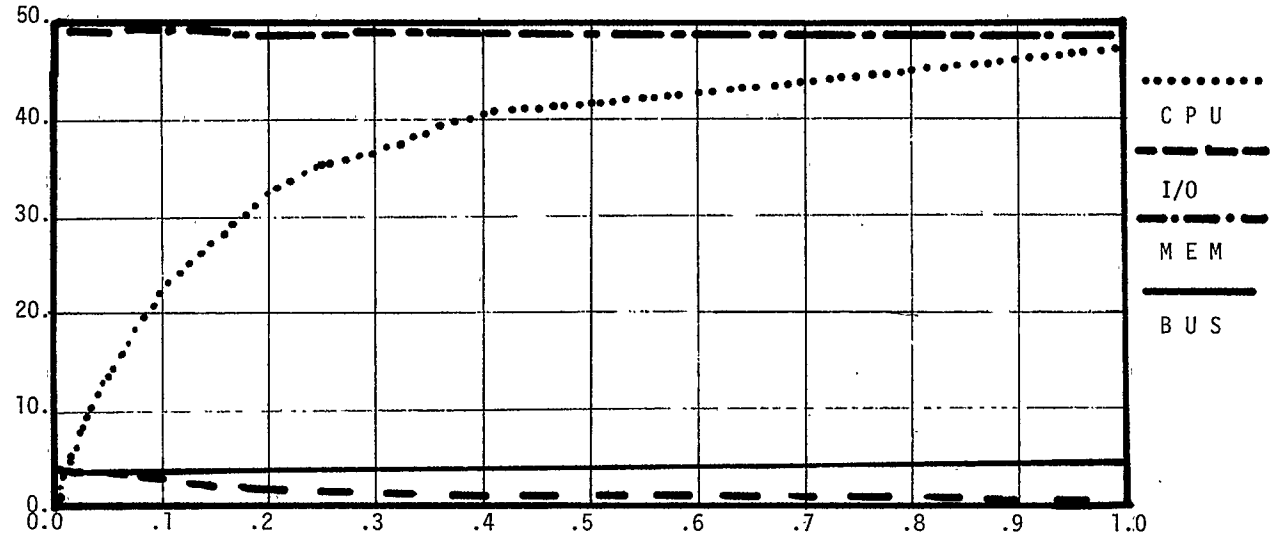

Fig. 6. ACTIVE Time versus I/O Frequency

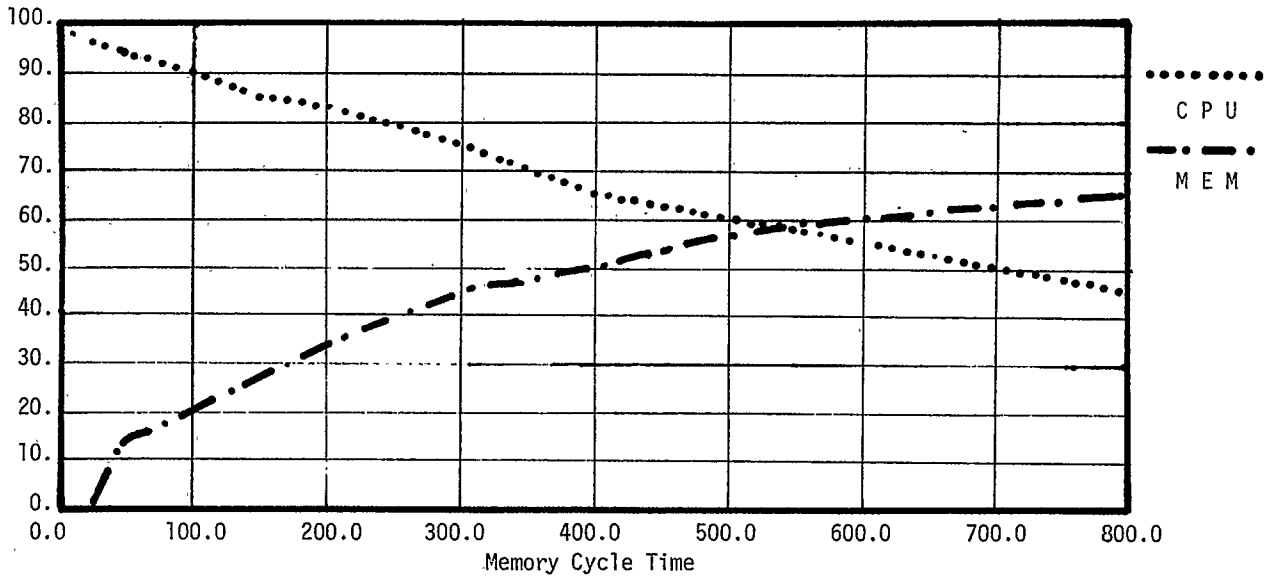

Fig. 7. ACTIVE Time versus Memory Speed

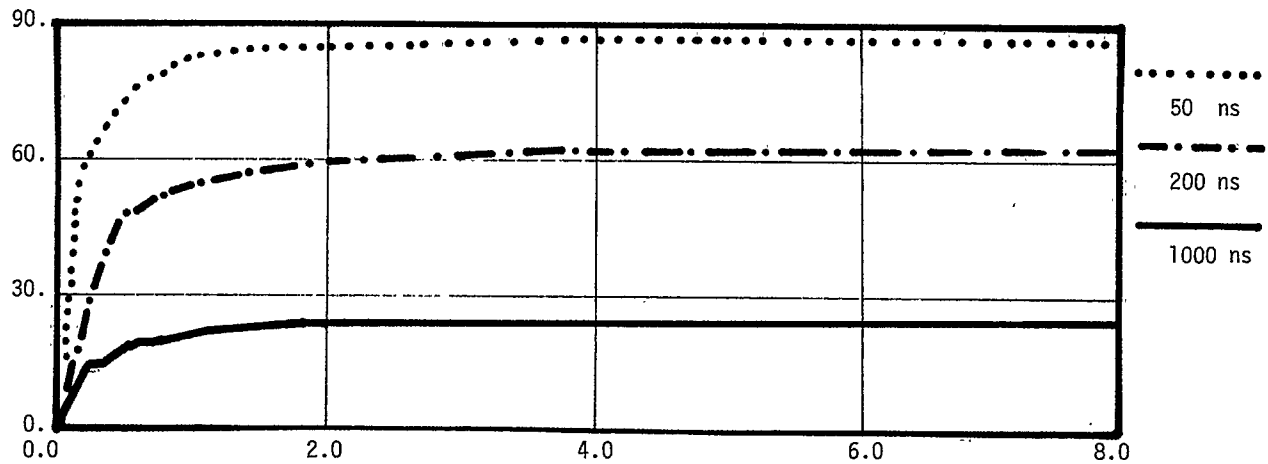

Fig. 8. C P U Utilization versus I/O Frequency

REFERENCES

Birtwistle, G. M. (1973), SIMULA BEGIN, Philadelphia, Auerbuck Publishers.

Birtwistle, G. M. (1981). A SYSTEM OF DISCRETE EVENT MODELLING ON SIMULA to be published.

Liebowitz, B. H., Multiple Processor Minicomputer System, COMPUTER DESIGN, Oct. 1978, pp 87-95.

Schriber, T. J. (1974), SIMULATION USING GPSS, John Wiley & Sons, New York.