GPSS SIMULATION OF A PROPOSED MINI-COMPUTER SYSTEM

Allen C. Schuermann and Everett L. Johnson
Wichita State University, Wichita, Kansas

INTRODUCTION

Many industries or design groups within a given industry, in an effort to minimize costs, view simulation as an expensive luxury not essential to the successful development of a product. The deluding aspect of this philosophy is that a series of design efforts may be successfully completed before a failure occurs which simulation might have avoided. The authors have participated in pathological simulations which were performed after the death of the project. These post mortems could have been avoided by early interaction between design groups and simulators. This paper discusses a successful interaction between design and simulation.

SIMULATION OBJECTIVE

The activities reported here are the result of a mini-computer design group realizing the potential value of simulation modeling in spotting troubles in the proposed design. Parallel efforts in the design and simulation of the new mini-computer architecture were initiated. The simulation effort was directed at developing a system level model of the mini-computer using whatever design specifications were fixed and treating the other specifications as parameters.

The amount of microcode executed per unit time was chosen as the standard of performance of the language processor(LP). Since one of the options was a multiple processor system, the number of processors which available memory access times and bus speeds would support was specified as a design parameter. Another design parameter of interest was the optimum size for cache memory. These parameters were varied to allow the designers to select values which met specified performance and economic criteria.

Another portion of the design problem was concerned with the input/output processor(IOP). It was necessary to determine a viable peripheral polling scheme and the number of IOPs required to perform in the specified I/O environment.

Since the language and I/O processor designs were to be completed in parallel, a natural division existed and two separate models were developed. The simulation data for the LP was considered most critical, so that model was developed first.

LANGUAGE PROCESSOR MODEL

The initial design of the system consisted of block flow charts which represented the sequential action of the system. Conservative best estimate times for various data transfers and overhead functions were included in the blocks. Other parameters which the designers felt were flexible and could be chosen to optimize performance were treated as parameters. Figure 1 provides the level of detail of the block diagram at which the simulation was performed.

Due to the block nature of the preliminary design and the similar block nature of GPSS, GPSS was chosen as the simulation language. The statistical functions available and the statistical data collection capabilities were powerful tools which made the modeling less complicated.

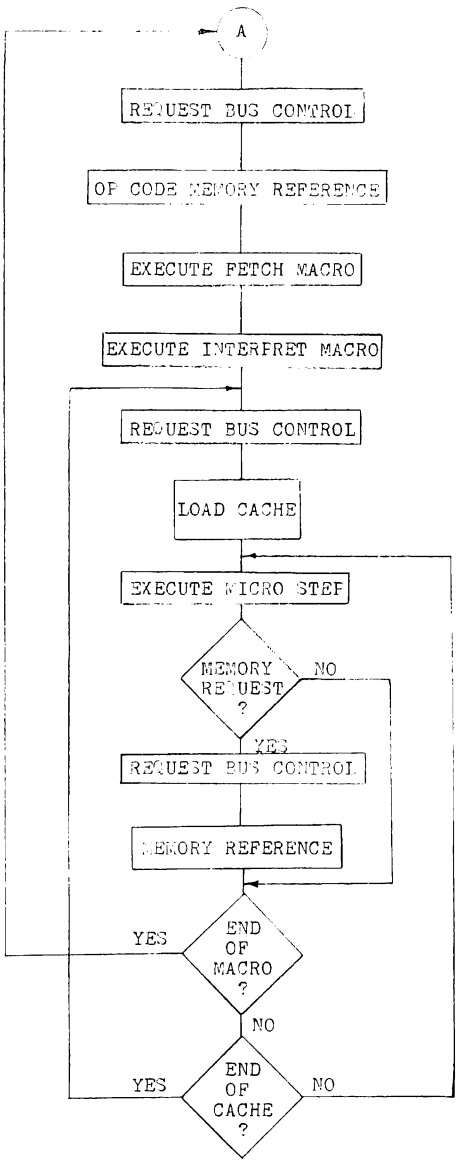

FIGURE 1 LANGUAGE PROCESSOR FLOWCHART

At this point, the exact I/O requirements were not known. Thus a statistical model of the I/O requirements was developed using typical peripheral data concerning lines of line printer output, CRT output and disk sectors accessed per CRT initiated activity. This statistical model was then used to compete with the LP for main memory.

The simulation was started by generating a single transaction which continued to request the loading of a new macro at the completion of each previous macro. The resultant microcode throughput reflected the maximum system performance with a single language processor. In order to determine the maximum throughput possible with multiple LPs, the unused memory access time was used in the same proportion to support additional language processors.

Simulation runs were made which simulated 1 millisecond of real time but took 20 seconds of CPU time on an IBM 360/44. Even though these runs were quite short, the simulation output was determined to be statistically significant. It should be noted that in the 1 millisecond simulation, the processor executed several hundred thousand microsteps.

Figure 2 presents the results of throughput versus macro length with cache size as a parameter. The following conclusions were drawn from the simulation results:

(1) If macro length is relatively fixed, cache size should be slightly larger than the macro length.

(2) A cache size of 16 words provided acceptable throughput for a wide distribution of macro lengths.

(3) Multiple language processors cannot be economically justified, since additional processors do not provide a proportionate increase in throughput.

I/O PROCESSOR MODEL

The I/O processor function involved two distinct processes - monitoring I/O initiating flags in main memory and polling peripherals to determine peripheral status. The flag reference task and the servicing of active peripherals during the polling exercise are impacted by contention with the language processor.

The following peripheral configuration was assumed for the simulation: 64 CRTs, 10 line printers, and 6 disks with an average of 12.1 lines of output on each CRT, 0.3 lines of output on each line printer, and 6.7 sectors accessed from the disk for each CRT initiated task.

The information required from the simulation was the number of I/O processors, time spent polling, time spent checking flags in memory, time elapsed between an I/O request and the beginning of service, time required to complete each I/O task, handshake response times, and the disk data buffer size.

The search of the I/O flag table in memory was initially simulated in detail with each entry being accessed and checked. In the event of an active flag, various pointers were accessed and examined. The time for each flag checked was used to advance the simulation clock. Similarly, the polling process was simulated to include the handshake time and whatever data transfer time was required if the polled device indicated service was needed.

Due to the short time required for each I/O device poll and I/O flag check, the GPSS program consisted of many short advance blocks. The initial detailed simulation run required 30 minutes of CPU time per second of real time. In order to reduce this time, a statistical estimate of the language processor contention with the I/O processor was developed.

Define p to be the probability that the main memory is busy and m to be the maximum duration of main memory usage by the LP. The probability distribution describing the waiting time R for the IOP to obtain main memory is

$$f(R) = \begin{cases} 1-p & R=0 \\ p/m & 0<R\leq m \end{cases}$$

The expected value of R, $E(R)=mp/2$. The variance of R, $Var(R)=m^2p(1/3-p/4)$. If the IOP requires n independent accesses to main memory, the total waiting time T will have the following characteristics:

$$T = \sum_{i}^{n} R_i,\ E(T) = nmp/2,\ Var(T) = nm^2p(1/3-p/4)$$

According to the central limit theorem, as $n \to \infty$ T is asymptotically normal. In fact, for the situation where p = 0.5 and m = 700 nanoseconds, T is approximately normally distributed for $n \geq 15$. Thus, if more than 15 main memory requests are required in succession, the main memory contention is approximated using $E(T) + z\{Var(T)\}^{1/2}$ where z is a standard normal random deviate. This procedure allows a series of memory contentions to be modeled as a single block rather than requiring the simulation of each individual memory access. When $n < 15$, a single advance was made, but with memory contention equal to n times the average contention time. Test runs validated the appropriateness of the above statistical approximation.

After the memory contention effects were approximated statistically, 10 minutes of CPU time were required per second of real time.

Further analysis now showed that the repeated polling and flag checking activity accounted for the major portion of the simulation activity. It was then realized that no IOP activity had to occur until a device became active or a flag was set. Thus, the simulation could proceed to the next active entry without intermediate polling steps. Since the time required to poll each device and check each flag was known, a function was derived and implemented which placed the polling or flag checking task at the proper location. Thus, the activity was picked up at the same instant in time as if the polling and flag checking had been going on continuously.

The simulation now required 10 minutes of CPU time to simulate 60 seconds of real time. Table 1 gives the improvements in run time as the simulation development proceeded.

TABLE 1

SIMULATION RUN TIME IMPROVEMENTS

	Simulation Time (CPU Minutes)	Simulated Time (Real Time Seconds)	Performance Ratio
Detailed Model	30	1	1800/1
Statistical Estimation of Memory Contention	10	1	600/1
Demand Responsive Polling	10	60	10/1

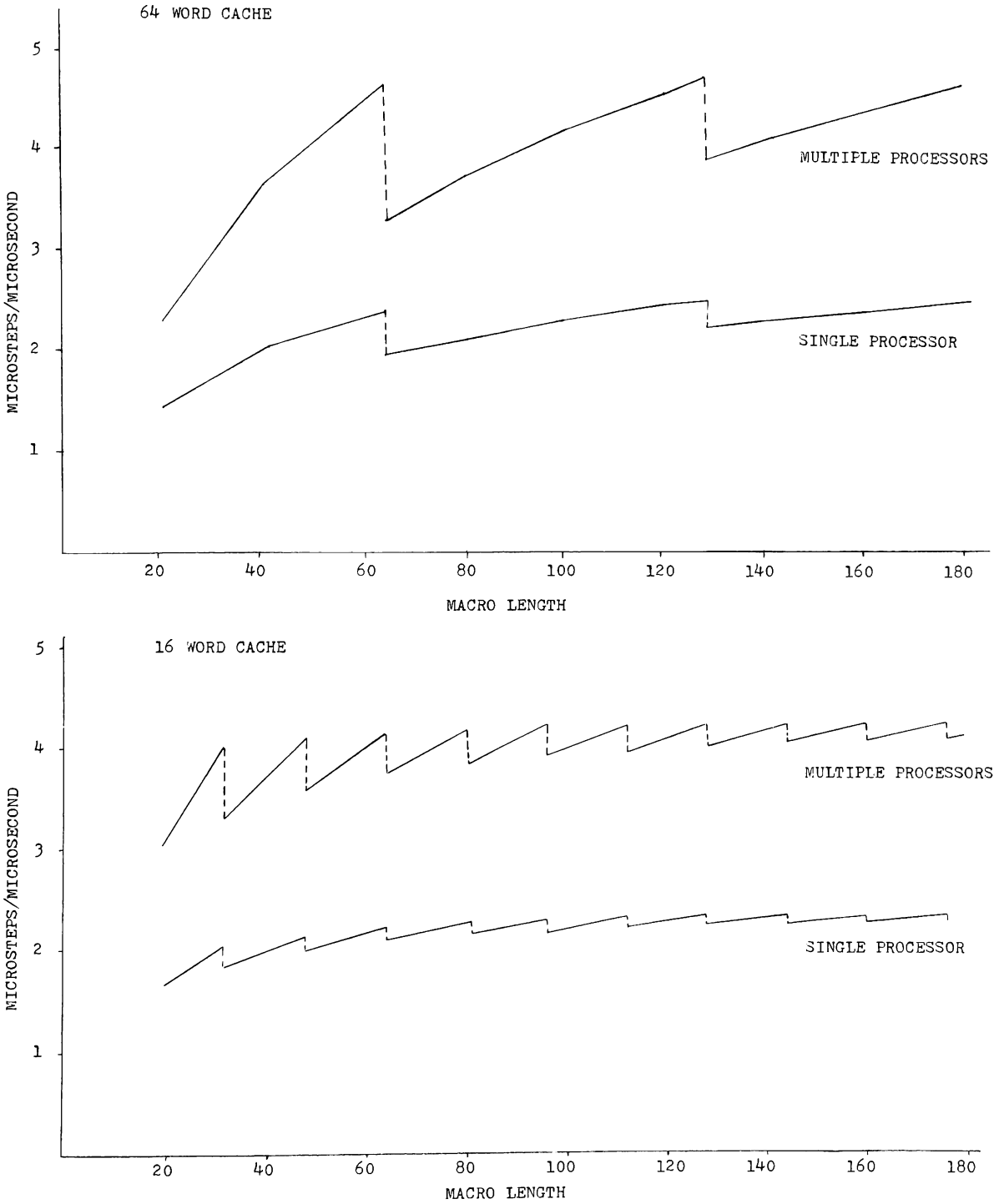

FIGURE 2 MICROCODE THROUGHPUT

The following benefits were obtained from the simulation model results:

(1) The model provided a means for recognizing and modifying inefficiencies in the flag checking algorithm.

(2) One IOP was sufficient to meet all performance specifications in the proposed I/O environment.

(3) Critical handshake response times were identified and their maximum allowable values determined.

SUMMARY

A successful parallel design and simulation effort has been reported which allowed management to make intelligent decisions regarding design directions. Classical steps in modeling development were used in constructing the simulation model:

(1) Detailed models were developed of the proposed design;

(2) The models were modified to include less detail, but still provided accurate external operation at significant savings.

ACKNOWLEDGMENTS

This research was supported by NCR Data Processing Division, Wichita, Kansas under contract #4405-50.

BIBLIOGRAPHY

1. Blakely, Max F. and Schuermann, Allen C., A Simulation Technique for Determination of Large Scale Computer System Channel Requirements, The Boeing Company Technical Document D3-8299, Wichita, Kansas, 1969.

2. Greenberg, Stanley, GPSS Primer, Wiley-Interscience, New York, 1972.

3. IBM Corporation, General Purpose Simulation System/360 OS and DOS Version 2 Users Manual, Second Edition, IBM Corporation, White Plains, New York, 1971.

4. Schriber, Thomas J., Simulation Using GPSS, John Wiley & Sons, New York, 1974.

5. Taha, Hamdy A., Operations Research: An Introduction, Macmillan, New York, 1974.

Allen C. Schuermann is an assistant professor of industrial engineering at Wichita State University. He received a B.A. in mathematics from the University of Kansas, an M.S. in mathematics from Wichita State University and the Ph.D. in operations research from the University of Arkansas. He previously held the position of Senior Operations Research Analyst with Boeing Computer Services. His current research interests include methods of evaluation for criminal justice and health care delivery systems and simulation methodology. He is a member of TIMS, ORSA, AIIE and Pi Mu Epsilon.

Everett L. Johnson is an associate professor of electrical engineering at Wichita State University. He received a B.S. in electrical engineering from the University of Kansas, an M.S. in electrical engineering from the University of New Mexico and the Ph.D. in electrical engineering from the University of Kansas. Previous positions were with the Boeing Company as a Research Specialist and Sandia Corporation. He has been a consultant to the Boeing Company and NCR. Current activities include microprocessor applications. He is a member of Eta Kappa Nu, Sigma Tau, Tau Beta Pi, ASEE and IEEE.