VLSI Circuit Design Supported By Computer System Simulation

John Nelson

Department of Electrical Engineering - Systems
University of Southern California[2]
Los Angeles, California

Abstract

Satisfying the enormous signal processing demands of space based sensors in the next decade is beyond the capability of integrated circuit technology as we know it today. However the performance promise of digital VLSICs (Very Large Scale Integrated Circuits) should make this problem soluble if the right device technologies, computational algorithms and computer architectures can be selected which best exploit VLSI for this application. In this space sensor signal processor design study, computer system simulation using the language ECSS (Extendable Computer System Simulator) proved to be the key to performing tradeoffs and developing designs which could meet the stressing demands for low power and high computational throughput.

INTRODUCTION

In order to assess digital signal processing technology as one of the necessary components of advanced space sensor systems planned for the late 1980s, DARPA and RADC sponsored the Advanced On-Board Signal Processor (AOSP) design study. The objective of the AOSP program has been to investigate future computer requirements for space based sensor systems and te determine whether an architecture or set of architectures can be developed for on-board signal processing using 1985 digital electronics which will meet the processing requirements and fit within the severe constraints of size, power consumption, reliability and radiation hardening imposed by the space environment. Additional goals of the AOSP architectures are (1) ease in software development, maintenance and adaptation to new applications and digital device developments, and (2) commonality of the architectural technology between various sensors and applications.

The trends in space sensor signal processing requirements of the 1980s (Table 5) will require logic throughput ranging above 1000 meg operations per second (add equivalent) and memory capacity up to 10,000 megabits. Table I contains projections of VLSIC device technology for this period which indicate dramatic improvements by 1985 in logic speed, complexity and memory density. The projections are based on an industry-wide survey and extrapolation of device improvements experienced in the last decade. The technologies considered are the most suitable mature semiconductor families from the standpoint of tolerance to the space radiation environment.

TABLE 1

PROJECTED 1985 SEMICONDUCTOR
DEVICE TECHNOLOGY PERFORMANCE

0.5 μm FEATURE SIZE TECHNOLOGIES	SINGLE GATE DELAY	1 CHIP MICROPROCESSOR			1 CHIP MEMORY		
	PS	BITS	MIPS	POWER WATTS	BITS	ACCESS NS	POWER WATTS
MOS	800	32	25	0.8	256K	100	0.05
BIPOLAR	130	16	250	1.5	32K	1	1.5
GAAS	75	4	500	0.1	16K	0.7	0.1

TABLE 2

SIGNIFICANT SIGNAL PROCESSING DESIGN OPTIONS

SYSTEM COMPONENT	DESIGN OPTIONS
VLSI LOGIC & MEMORY CIRCUIT TECHNOLOGY	SUBMICRON CMOS/SOS, BIPLOAR, GALLIUM ARSENIDE
SIGNAL PROCESSING ALGORITHMS	SPECTRAL ESTIMATION TRANSFORMS: FFT, WINOGRAD, DFT; DIGITAL FILTERS; EDGE DETECTORS; IMAGE PROCESSING TRANSFORMS
COMPUTING SYSTEMS ARCHITECTURE	PIPELINING; LOOSE & TIGHT COUPLING PARALLEL; MEMORY HIERARCHIES

[1]This work was performed as a portion of contract F30602-78-C-0028, "Advanced On Board Signal Processor Technology" and simulation under the supervision of John V. MacNamara, Rome Air Development Center, Griffiss Air Force Base, Rome New York, and was also sponsored by the Defence Advanced Research Programs Agency,Strategic Technology Office.

[2]Work reported in this paper was performed previously as an employee of the Hughes Aircraft Co., Culver City, California.

Given the semiconductor technologies and the processing requirements of the space sensor applications, the problem then became one of selecting computer system designs and algorithms to accomplish the signal processing. Because the VLSI technologies offer new opportunities for computer system architectures and for the application of new algorithms [4] a wide variety of designs had to be considered.

This challenge demanded a novel approach for the rapid performance evaluation of algorithms and architectures. Since no hardware could be fabricated now using 1985 IC technology, simulation turned out to be the only meaningful approach for design modeling and evaluation. Because of the real-time nature of signal processing the computational throughput for the required algorithms is the primary measure of performance. Thus much consideration was given to the selection of an appropriate method of simulation which could accurately reflect the performance of signal processing algorithms for a variety of optional computing system architectures.

Many possible levels of computer technology modeling and simulation detail can be considered from discrete semiconductor devices (circuit-level) to complex processors (processor-memory-switch, PMS, level) [2]. Consideration had to be given to the great expense entailed in circuit level simulation as against the need to evaluate overall PMS level performance while trading off competing semiconductor technologies. The challenge was to reflect sufficiently detailed characteristics of semiconductor technology (e.g., CMOS versus ECL) in order to make performance evaluations of entire computing systems composed of any candidate technology.

Four typical signal processing applications were considered:

1. low-resolution radar
2. high-resolution radar
3. narrow coverage electro-optical
4. wide-coverage electro-optical

In order to alleviate much of the burden of simulation and tradeoff, a number of preliminary decisions were made about the set of design options which should be simulated. Thus it was decided that general purpose architectures should be considered rather than algorithm specific ones because of the need to use the designs for a wide variety of sensors and to carry out many types of processing, varying from spectral estimation to pattern recognition. In addition it was immediately apparent that no one serial processor similar in architecture to current commercial types would be adequate for this application. Evidently some type of parallel processing would have to be used to provide the high throughput rates required since semiconductor device improvements alone could not do the job.

SELECTION OF A SIMULATION METHOD

The design options that have most significant effect on the overall computing system throughput performance for this space sensor application appear in Table 2. It is essential to model and evaluate these options by simulation. When these design options are chosen and incorporated into a simulation model, the model must reflect those effects in the interaction of these choices which contribute to overall computational throughput. There are a variety of well-known effects that limit the efficiency of parallel architectures, including resource contention and a lack of parallelism in algorithms. A suitable simulation method must realistically model these effects. In addition the simulation should satisfy a number of general criteria which appear in Table 3.

TABLE 3

CRITERIA FOR DEVELOPING AOSP SIMULATION

1. Easy modification for tradeoffs
   - change algorithms
   - change system architecture
   - change logic and memory speeds
2. Accurate representation of hardware constraints
   - memory capacity
   - bus capacity
   - arithmetic/logic speed
3. Realistic modeling of multiprocessor phenomena
   - resource contention
   - communication delays
   - algorithm "fit"
4. Good statistics
   - throughput
   - queues, bottlenecks
   - resource utilization

ECSS AS BASIS FOR ARCHITECTURAL TRADEOFFS

The general purpose comptuer system simulation language ECSS (Extendable Computer System Simulator) was selected for AOSP architectural modeling, performance evaluation and design tradeoffs. ECSS II is a specialized language for constructing discrete event simulation models of computer systems. It provides statements for describing common computer hardware configurations, software operations, and workload characteristics in a natural and straightforward notation. Using these statements, one can compactly express, for example, the name, quantity, and performance of each kind of simulated hardware device, the behavior and resource requirements of each kind of job or algorithm to be processed, the policies by which resources are assigned to jobs, the characteristics of messages sent through I/O devices within the model, and how the simulated system is to be loaded by jobs and messages from its environment [3].

As a flexible way of intergrating these statements together to form a particular model, ECSS provides a full set of lower-level commands commonly found in less specialized programming languages: branching, looping list-processing, input/output, computation, and so on. These capabilities are supplied by including the general purpose simulation language SIMSCRIPT II as a subset [3].

These features satisfy the above criteria for flexibility, modeling realism, and performance statistics collection. A partial list of the models in the language which allow direct representation of the AOSP design is contained in Table 4. In general the language proved appropriate and adequate for handing the design parameters (device characteristics, architectural components and organization, algorithms) and evaluating the performance of a variety of architectures (throughput, duty factor).

TABLE 4

ARCHITECTURE DESIGN PARAMETER MODELING IN ECSS

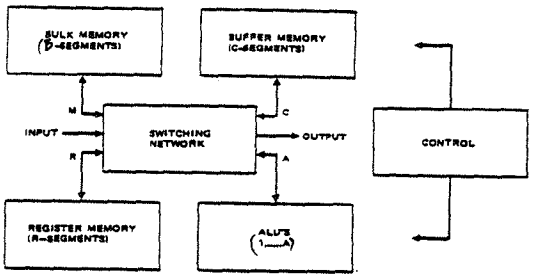
DESIGN PARAMETER	CHARACTERISTICS	ECSS MODEL
DEVICE FAMILY	LOGIC SPEED; MEMORY SPEED, CAPACITY	EXECUTES 10^7 ADDS/SEC, TRANSFERS 10^6 WORDS/SEC, STORES 10^5 WORDS
MULTIPROCESSORS	NUMBER, CLOCK	SPECIFY 100 CPUs WHICH EXECUTE 10^7 INSTRUCTIONS/SEC
COUPLING	LOOSE, TIGHT	PATH p CONNECTS CPU1 TO CPU2
OPERATING SYSTEM	RESOURCE ALLOCATION, SYNCHRONIZATION	ALLOCATION MANAGERS WAIT, SIGNAL
ALGORITHMS	TIME SEQUENCE OF STORAGE & DATA PROCESSING OPERATIONS	EXECUTE 5 ADD INSTRUCTIONS

SIMULATION MODELS

The models selected were quite general, permitting representation of a broad range of architectures. Figure 1 shows the general architectural model

FIGURE 1

GENERAL AOSP ARCHITECTURAL MODEL
USED IN SIMULATION TRADEOFFS



used in ECSS simulations. The four basic hardware resources are bulk memory, buffer memory, register memory, ALUs and an interconnection or switching network. Bulk memory is characteristically large in capacity but has a shorter access time than bulk memory. Buffer memory provides a random access bandwidth comparable to a register memory segment bandwidth over a short period (less than a 0.5 duty factor). Register memory is usually smaller in capacity than buffer memory and has a cycle time less than or equal to the shortest execution time of any ALU operation (addition). The ALUs are capable of performing the basic arithmetic and logical operations required for signal processing including fixed point addition, multiplication, subtraction, and right and left shift. The following architecture parameters were varied to describe a large variety of specific architectures.

1. Number of segments in the bulk, buffer and register memory devices.
2. Storage capacity of each segment of the memory devices.
3. Bandwidth of each segment of the memory devices.
4. Number of ALUs.
5. Operation rates of each ALU.
6. Number of channels connecting each device to the switching network.
7. Switching capabilities of the network.
8. Transmission capacities of the links between switches in the network.

For the prupose of AOSP architectural tradeoffs, the control models available in the ECSS service routines were utilized to provide:

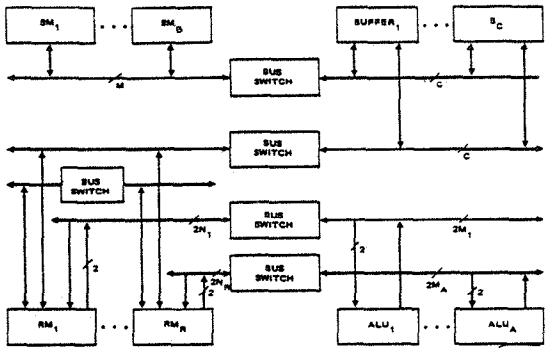
1. communication paths in the switching network as requested by the job stream.
2. storage allocation as requested.
3. arithmetic processing as requested.

4. queueing of requests which cannot be immediately satisfied.
5. Synchronization of concurrent processes as specifice in the workload.

Figure 2 shows how particular choices of interconnection switching networks yield a broad range of specific architectures.

FIGURE 2

COUPLED CPU ARCHITECTURE



ARCHITECTURAL TRADEOFF PROCEDURE

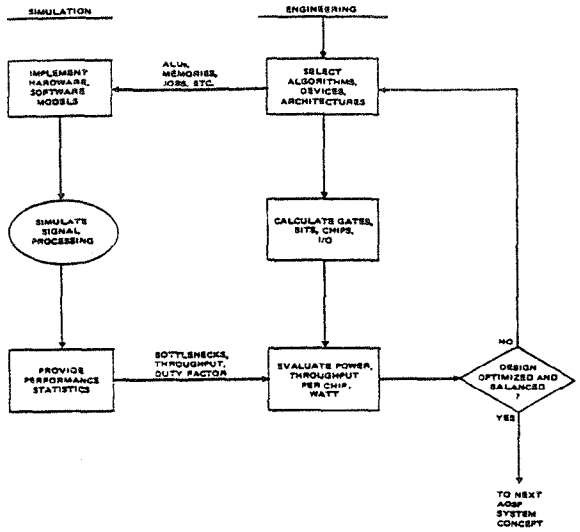
The procedure used to conduct quantitative tradeoffs of architectures using simulation is depicted in Figure 3. The parameters selected as input to the architectural model presented in Figure 1 were selected from the estimates of VLSI device characteristics for 1985. The speed of logic functions such as ALUs is calculated from known designs for each concept (e.g., array multiplier) to be traded off. The speed is then reflected in the simulator by a statement defining speed of the ALU. For example:

"SPECIFY 4 ALUs, EACH EXECUTES 1 MULTIPLY INSTRUCTION/CLOCK CYCLE"

This is done analogously for memories, busses, and other elements whose speed and/or storage capacity is a function of the device technology chosen. The algorithms to be performed are reflected as jobs which require some sequence of arithmetic and data handling operations to be carried out by the architecture. Once the architecture (system description) operating system (ECSS service routines) and the algorithms (workload) were modeled in ECSS,the simulation was carried out by translation of the models into host computer assembly language (i.e., Amdahl 470) and execution of the resulting code.

FIGURE 3

ARCHITECTURAL TRADEOFF PROCESS



The extensive performance statistics generated by ECSS provided feedback to the engineering tradeoff process. The ultimate evaluation of the architecture efficiency is made by computing the number of gates required for all logic functions including control in the architecture tested, the number of bits of memory required for storage and partitioning these gates onto chips of a complexity corresponding to the device technology simulated. The power dissipationof each chip together with its duty factor and the overall throughput provided by simulation allowed the throughput per watt and total power to be calculated. The imbalance of computational,storage and communication resources reflected by their duty factors and the processing bottlenecks reflected by the buildup of queues were quite apparent from the performance statistics. These indications of architectural inefficiency guided the modification of the architecture for improvement.

The following approach was employed to optimize the signal processor architecture for four typical electro-optical and radar systems which are representative of future space sensors.

Step 1: Optimizing a Single Processing Element

Starting from a minimal processor (roughly equivalent to existing microprocessors) with adequate data word size (16 bits) various architectural features which take advantage of processing operation parallelism

and repetition were evaluated. They were incorporated in the processing element architecture only if they provided better than linear speed-up (N processors yield N times the throughput) in processing per gate expenditure for their implementation. If the added features yielded less than linear speed-up they were rejected. Thus, the processing element was optimized for throughput per gate subject to the I/O (pin or pad) limitations of the device technologies.

Step 2: Optimizing the System Architecture

Starting with an optimized processing element from step 1, arrays of multiple processors were evaluated and optimized on the basis of throughput per watt subject to the constraint that the minimum required throughput for each signal processing system was attained. The architectural options evaluated included variations in the

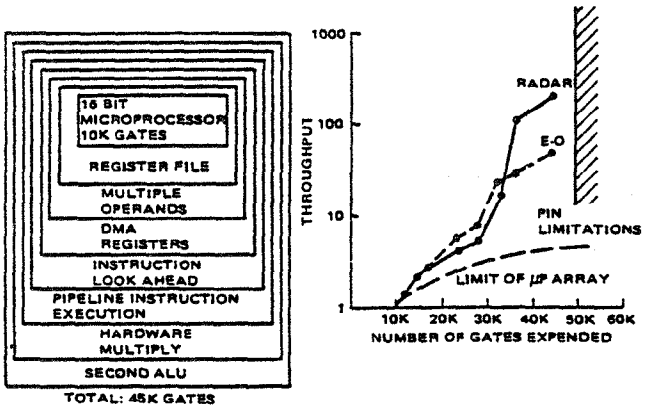
1. number of processing elements.
2. communication network between processors and memories.
3. levels and segmentation of the memory hierarchy.

SIMULATION TRADEOFF RESULTS

The single processor optimization procedure described in the preceding section (step 1) is the incorporation of architectural features in a minimal microprocessor based on their ability to deliver more throughput than can be expected from an array of multiple processors. This tradeoff process and the results are depicted in Figure 4.

FIGURE 4

OPTIMIZING PROCESSING ELEMENT ARCHITECTURE



The smallest rectangle to the left indicates a single chip microprocessor with roughtly the functional capability of a Motorola 6800 expanded to a 16 bit data word size. The features added to take advantage of parallelism and repetition in the processing load for the radar and electro-optical concepts are indicated by an increasing sequence of rectangles surrounding the original microprocessor. The total number of gates finally added to incorporate these features is plotted on the right to indicate its cost in gates and the resulting throughput improvement relative to the throughput of a single processor.

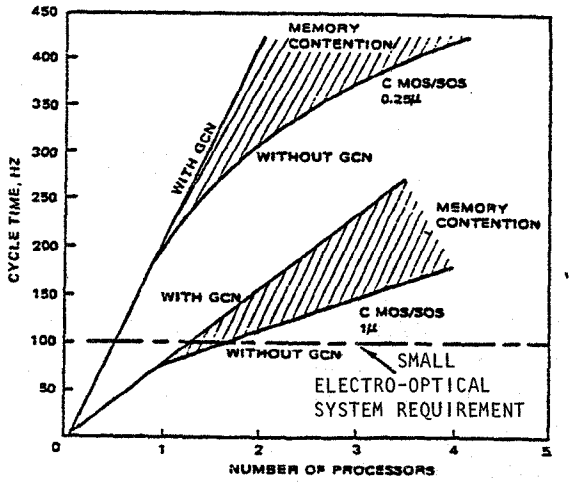
The dashed line on the plot indicates a linear upper limit on the throughput which could be expected from an array of microprocessors with 100% efficiency. Other features considered which could have been incorporated in the original processor resulted in either a less than linear throughput improvement or could not be used to advantage due to the ultimate I/O constraints of the technology, assumed to be hybrid packaging with a limit of 300 pads per chip.

SYSTEM ARCHITECTURE OPTIMIZATION

A variety of combinations were investigated in the optimization of the system architecture. One of the most significant results, see Figure 5, shows that unless adequate communication is provided between the ALUs and buffer memories (see the model of the previous section) less than linear throughput improvement is obtained from an array of optimized processing elements. By using a generalized connection network (GCN) to enhance communication, this bottleneck is alleviated.

FIGURE 5

MEMORY CONTENTION IS REDUCED BY GCN

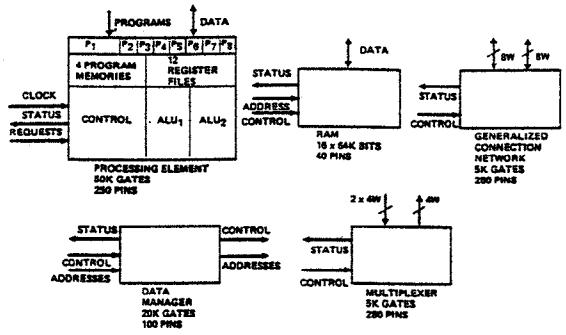


COMMON AOSP CHIP SETS

The design optimization studies described previously resulted in a preliminary definition of a common set of chips for the four system concepts investigated. These are presented in Figure 6. The chip set conclusions are for the most part functionally independent of the device technology. However their realization in specific technologies will involve architectural bit slicing in some cases and multiprocessor chips in others depending on device complexity.

FIGURE 6

COMMON AOSP CHIP SET



A large variety of device technology combinations were considered in the AOSP design tradeoffs and each resulted in a different optimum system architecture for a given application of the four considered. However, the optimum processing element architecture was substantially the same for these applications, the difference being the number of processing elements, the interprocessor communication bandwidth, the size of the bulk memory and the need for a buffer memory in the memory hierarchy. A list of efficient architectures for each application and their characteristics appears in Table 5.

TABLE 5

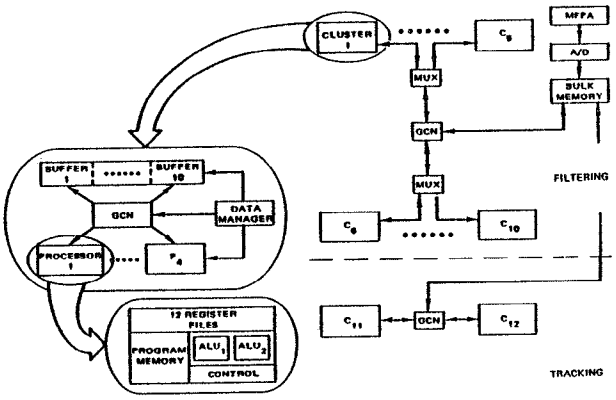
SIMULATION TRADEOFF RESULTS

MISSION & SENSOR	REQUIRED THROUGHPUT (OPS/SEC)	TECHNOLOGY	NUMBER OF PROCESSORS	LOGIC CHIPS	MEMORY CHIPS	POWER (WATTS)	5-YEAR RELIABILITY
LOW RESOLUTION SPACE RADAR	1.0×10^8	1µm CMOS/SOS	1	3	9	2	>0.9999
HIGH RESOLUTION SPACE RADAR	6.4×10^8	1µm BIPOLAR 1µm CMOS/SOS	1	21	73	9	>0.9999
NARROW COVERAGE SPACE ELECTRO-OPTICAL	1.8×10^8	1µm CMOS/SOS	2	10	430	16	>0.9999
WIDE COVERAGE SPACE ELECTRO-OPTICAL	8.3×10^{10}	0.25µm BIPOLAR 0.25µm CMOS/SOS	48	536	17,000	257	0.9999

A block diagram of a high performance architecture associated with a wide coverage electro-optical sensor which utilizes the standard chip set is depicted in Figure 7.

FIGURE 7

WIDE-COVERAGE ELECTRO-OPTICAL SIGNAL PROCESSOR ARCHITECTURE



CONCLUSIONS

Designing VLSI architectures promises to be a rewarding challenge because of the complexity and performance projections for future semiconductor technologies. This design study required a look into the future at a very demanding application for VLSICs. Because a number of design options appeared promising and quantitative performance tradeoffs were required at diverse levels of design detail, traditional methods of breadboarding and detailed simulation were inappropriate. The discrete event simulation language ECSS/SIMSCRIPT with its special constructs for representing computing system components proved the best tool for multi-level modeling, design evaluation and tradeoffs. The general nature of the language facilitated representation of a system level components such as a memory while at the same time permitting the modeling of more detailed entities such as a high speed adder. The extendable nature of the language accommodated the modeling of a parallel processing executive which could allocate computing resources to concurrent processes dynamically. The built-in model operating system provided by the language greatly reduced the programming effort required. In general the language facilitated the modeling of extremely diverse computing system components and the evaluation of a broad range of algorithms.

The ECSS language has been traditionally applied to modeling central processors, memories and peripheral components. This study showed that the language could be effectively applied to model virtually any computing system component or computational operation. However some extensions to ECSS were required in the form of SIMSCRIPT programs to model, for example, the operation of an ALU. As a result of this study is apparent that ECSS can be a very useful tool in VLSI design tradeoff studies in the future and it can be greatly enhanced for this purpose by the addition of built-in models to handle parallel processes and relatively low level design primitives.

ACKNOWLEDGEMENT

I would like to acknowledge the assistance and helpful guidance of my former colleagues at the Hughes Aircraft Company, Dr. Howard Baller and Frederick Dellinger in the signal processor study. Also, an invaluable introduction to ECSS principle and practice was provided by Dr. Yen W. Chao of the Federal Simulation Agency, and by his associates, especially including Lt. Peter Hsu.

REFERENCES

1. Boehm, B. et al., "Simulation Aids for Designing Integerated Information Systems: the ECSS Language," Astronautics and Aeronautics, Nov. 1972.

2. Breuer, Melvin (ed.), Digital System Design Automation: Languages, Simulation and Data Bases, Woodland Hills, California, Computer Science Press, Inc., 1975.

3. Neilsen, N.T., "ECSS: Extendable Computer System Simulator," Santa Monica, California, The RAND Corporation, RM-6132-PR, Jan. 1970.

4. Reed, I.S. and Truong, T.K., "A New Hybrid Algorithm for Computing a Fast Discrete Fourier Transform," IEEE Transactions on Computers, vol. C-28, no. 7, July 1979.

FROM THE EDITOR'S DESK

(continued from page 5)

members of our 2,000 membership were to write a single article, we would be well stocked for a few years. Why don't you do your share in keeping articles flowing?

Please, if you plan to send in announcements of future meetings or a call for papers, recognize that it takes about six weeks from the time I start to put an issue together until it is received by the members (and that only if the US Mail gets through). Few conferences are planned at the last minute, so in the future please drop me a note even if full literature has not been prepared.

From now on you will be receiving SIMULETTER on a regularly quarterly basis. The next issue should reach you in June and the one after that in August. In that way we shall be back on schedule.

HELP WANTED

In an earlier issue I had asked for volunteers to be guest editor of a special edition of SIMULETTER. Again, I make this appeal to our readers.

I have had several members talk with me about doing a special topic issue and I hope that these will materialize over the next year or so. It is this way that we can obtain some unusual and interesting issues in the future.

Thus far I have not "put pressure" on anyone. I have considered noting the individual's name and possible topic which can be prepared. For example, how about a special issue on:

- statistical validation of simulation models
- energy models
- financial and business models

These is one one discrete simulation languages which is being prepared by your editor for late in 1980. It will require cooperation from a number of individuals and several have already offered their services. If you would like to join in preparing this late 1980 issue, please write to me.

(continued on page 32)