THE APPLICATION OF ASSOCIATIVE PROCESSING IN DISCRETE SIMULATION

Jeffrey L. Posdamer*
Systems and Information Science

Robert G. Sargent**
and
P. Bruce Berra**
Department of Industrial Engineering
and
Systems and Information Science
Syracuse University
Syracuse, N.Y. 13210

Abstract

The associative memory is a digital storage device in which all words can be simultaneously accessed and compared to a search argument. The associative processor is an extension of the associative memory which adds the capability for arithmetic operations within all words simultaneously. This paper will discuss the effects of the associative processor on each of the steps of discrete simulation. Specific topics discussed will include model construction, programming, running the simulation and data analysis.

Introduction

The stored program digital computer has existed in its present form for approximately twenty-five years.[9] In this period, the organization of the computer has remained essentially unchanged. In particular, it has been a serial processing unit which addresses both data and instructions by physical location in memory.[1]

Thus, all problems, regardless of their structure, must be mapped into a serial solution form. A more appropriate approach would be the choosing of a processor structure which matches the problem. This has been impossible due to the lack of variety in available machine structures.[11]

Several other approaches to organizing information processing systems have been investigated. One approach which has been studied is that of concurrent or parallel operation upon many data items. This approach reduces the mean time of execution by at least a factor of the reciprocal of the number of data items involved. Another technique which has been investigated is that of accessing data by content rather than by physical location. The combination of these two techniques appears in a device known as an associative processor.

The structure of most simulation problems has aspects which indicate that the usage of an associative processor may be effective. As a data management problem, the work of building, accessing and manipulating data structures will be enhanced by use of content addressability. As a description of processes which occur simultaneously both efficiency and understanding will be improved by the availability of parallel processing. The match between problem structure and machine structure will y ld not only substantive improvements in performance but a simplification in understanding generated by the removal of a great deal of detail needed to convert from a parallel problem to a serial solution.

After discussing the operation of the associative processor, the impact of content addressability and parallel search will be considered with respect to the steps of model formulation, programming, running the model, and data analysis.

Associative Memories/Processors[14]

An associative memory is a hardware peripheral device which operates in conjunction with a sequential computer. Data are stored in fixed length words as in sequential memories but are retrieved by content rather than hardware storage address. When a word or a portion of a word is desired the entire contents of the memory are searched in one memory access thus resulting in very fast search times. In addition to such searching criteria as equal, maximum, between limits, etc. associative memories can perform Boolean operations. If an associative memory possesses arithmetic capabilities it is called an associative processor.

In this section associative memories and associative processors are discussed and some application areas referenced.

Figure 1 indicates the layout of an associative memory. The data are stored in parallel fixed length words in memory and can be divided into any desired bit fields as shown by $F_1, F_2, \ldots, F_n$. The Comparand and the Mask Registers are utilized in searching the memory for desired data. The Response Store is utilized to indicate the results of a memory search. For example, suppose one desires to search the first n bit positions of all words in memory for an exact match with an external word. The search proceeds as follows. The external word is placed in the Comparand Register. The Mask Register is then utilized to mask out any undesired bit positions (those above n). This is done by placing ones in the first n bit positions and zeros in the rest. In this way only the desired bit positions are utilized in the search. In performing the search, the operation takes place in a word parallel - bit

serial fashion. That is, the first bit slice across all words in searched simultaneously followed by the next bit slice, and so on until all desired bit slices are searched. If any word(s) in the memory match the word in the Comparand Register a response bit is set in the Response Store. Then the Response Store can be searched and the desired words accessed. Thus, any bit position or combination of bit positions serves as an address to be utilized in searching operations and this is the so called "content addressability" property.

With the capabilities indicated above the associative memory can be utilized in:

a. Retrieving all words in which a specified field satisfies a specified search criterion. These criteria include equality, inequality, maximum, minimum, greater than, greater than or equal, less than, less than or equal, between limits, next higher and next lower.
b. Forming complex queries by logically combining the above criteria. Boolean connectives include AND, inclusive OR, exclusive OR and complement.
c. Defining any number of fields within the word with no conceptual or practical increase in complexity.

In addition to the capabilities already mentioned, associative memories can be constructed to have the ability to perform arithmetic operations simultaneously between the fields of every selected word. Devices of this type are called associative processors.

Operations that can be performed on all words that satisfy some specified search criteria as previsouly described include:

a. Add or subtract a constant relative to a given field.
b. Add, subtract, multiply or divide two fields and place the result in a third field.

This additional capability extends the use of the associative processor to a large class of scientific problems in which similar operations are repeated for a multiplicity of operands.

Associative memories and processors find application in a wide variety of fields. Some of these fields include Air Traffic Control,[10] Criminal Identification,[5] Operations Research Networks,[8] Computer Graphics,[12] and Data Management Systems.[2]

The Effects of Associative Processing

The various steps of the discrete simulation process will now be examined with respect to content addressable searching and parallel processing. While the following discussion relates the advantages of using a full scale associative processor, some of these advantages could still be obtained by use of a subset of its properties.

In the context of simulation the term programming takes on two meanings. The first or high-level meaning is that associated with the users of simulation systems, that is those who program in high-level languages, such as SIMSCRIPT or GPSS. The second or low-level meaning is that associated with the design and implementation of simulation systems. This is the level of programming using general purpose languages, particularly assemblers, to build simulation language processors, model running executives, report generators, etc. It is clear that there exists a continuum between the above definitions and that much work falls somewhere in the interval between high and low level.

To the user of a higher level language, the impact of the associative processor will be less apparent than to those operating at the lower levels. One reason for this is that all of the products of current computer organizational schemes have, implicit in them, the serial nature of the machines upon which they were implemented. This implicit structure may mean that current systems will be adapted to associative processing only with difficulty.

Performance improvements will be observed in the processing of high-level languages for which compilers have been implemented on an associative processor. A large portion of the language processing task is the building of tables of data, searching and accessing these tables to build the model and inputs.[7] These tasks are particularly well suited to the use of content addressable searching. Additionally, the relationships between elements to be indicated in the model's data structure may be more easily shown in an associative data structure. The path of flow of entities through the model may be recorded by indicating in each process description the predecessors of that process. By use of this technique an entity can move to all successors by use of an associative search based on the name of the entity generating process. A final saving would be the freedom from data ordering and/or contiguity and its resultant storage management problems which are eliminated in an associative processor.

As was indicated earlier, present simulation systems do not easily lend themselves to taking advantage of the associative processors structure. As a result of this, it is expected that new simulation systems will eventually arise as a result of the new hardware. The direction that these systems will take is unclear, but will emphasize the parallelism of associative processing.

The effect of content addressable memory on the low-level user is more direct. This is because he directly sees the structure of the machine. As he generates his simulation language processor the improvements in language processing take place. The availability of the hardware associative memory allows the efficient implementation of associative data structures.[3] Such structures simplify the laying out of the output of language processing, that is the data describing the processes and the entities upon which they act.

In the direction below the simple queueing model given in Figure 2 will be used. The model includes two independent servers, A and B, each fed by an independent arrival process stream. The output of A and B is fed into another independent server, C. The queue discipline for each server is first come,

first served, with an infinite queue allowed (Q_A, Q_B, and Q_C).

To see the effects of content addressability it is necessary to examine the data structure characteristics of the model. Each process would be represented as an identifier, table of predecessors, and a set of functional parameters, as shown in Figure 3(a). The entities which are the objects of the processes consist of the name of the currently controlling process, name of the entity and parameters as shown in Figure 3(b). The individual data items thus directly contains relational data not internal names (e.g. indices or pointers). An entity can be transferred to a process by changing the appropriate field in the data item. To move an entity from Q_A to Q_C it is only necessary to change the controlling process from A to C. No mention of item order is made intentionally. Such ordering is not needed. It should also be noted that since the data may use the actual item names it is more closely related to the original model description. Since relational data are reduced in complexity and certain control items are eliminated, smaller data structures are likely to result. A reduction in data size can yield two benefits. Larger models may be represented in the same storage size or alternatively, several variations or perturbations of a model may be represented simultaneously.

The application of the content addressable search to the running of the simulation can be seen from some simple examples. To propagate an entity to the processes fed by an individual process a content addressable access can be made to memory in which the predecessors of processes are checked against the current process. This yields all successors in one search. In terms of the example shown in Figure 2, to find all the successors to process A a search is made for processes of which A is the predecessor. This yields C as the only successor. To find all entities currently assigned to a process, a search would be made of all entities having the process name in the "controlling process field". This simplified searching of lists would speed up the execution of the running of the model.

In the final stages of analysis and report generation the ability to select data by content will also have some effects. Most obvious is the lack of need to sort data for analysis or reporting. Data in an associative memory can be accessed in sort order regardless of physical order. An advantage for report generation is the presence of actual names in the data structure which do not have to be converted to be understandable as do indices or pointers.

The second important property of the associative processor is that of the ability to do parallel processing. This property gives both qualitative and quantitative improvements to the simulator. Most simulation models contain processes which run simultaneously. On conventional computers this parallelism must be broken down to sequential operations. On the associative processor those processes which should run in parallel may do so.* This simplifies both the understanding of how the model is to operate and the coding of the model. This ability to model parallelism will undoubtedly yield new techniques of model formulation and new simulation languages which take advantage of its power.

In the actual running of the model speed improvements will result from parallel processing. Since several (or all) processes may be evaluated simultaneously a saving of at least the reciprocal of the number of simultaneously executed processes will be realized. Greater savings may result as the need for various control structures (DO-loops, etc.) is eliminated. Problems which now may arise due to the sequential processing of supposedly parallel processes will likewise be eliminated. Thus during the running of the model large savings in time should be realized.

The capability of running several models simultaneously provides several new options to the simulation user. If n identical models are run simultaneously with each having its own independent random number generator, n independent sets of output data or realizations are obtained. In a conventional computer, a user would probably use blocking techniques or make n simulation runs to obtain the n independent observations. Thus, the associative processor allows efficient calculation of ensemble averages for non-stationary, transient, or steady state responses. Another use of independent observations generated in this way is in statistical analysis techniques such as analysis of variance or multiple ranking tests. A user also has the option of choosing n different initial conditions or n different parameter values in the n models to determine the effects of various initial conditions or the sensitivity of the model's response to its parameters, i.e. sensitivity analysis. If strategies representable by different sets of parameters are used one could use the same random variates for equivalent events in the different models resulting in positive correlation between the results of the differing strategies. This, of course, would reduce the variance between the different strategies for a given computer run time.

In using the associative processor to analyse data, the advantages are less clear. Feng[6] has shown that one can generate Fast Fourier Transforms more efficiently. This indicates that spectral analysis could be done more expeditiously.

*More properly, all processes whose outputs cannot effect other processes currently being evaluated within the time increment used for stepping this model may be executed simultaneously. The problem of time incremented vs. event incremented processing is open to research in this area.

The advantages, if any, in using the associative processor for computations of analysis of variance, multiple comparison tests, and multiple ranking procedures requires research.

Search methods or experimental designs such as factor analysis and evolutionary operations (EVOP) are normally used in optimizing simulation models.[4] Experimental designs usually require independent observations. These can be generated efficiently by running several models simultaneously as explained above. If search methods are used, a choice is usually make between simultaneous or sequential search.[13] Sequential search is more efficient when one observation at a time is obtained, as is the case with the conventional computer. With an associative processor, simultaneous search can be performed when running models in parallel. This additional option will require a user to make new decisions as to which search method is the most efficient to use. In searching multi-model surfaces, a given search method using n different starting points can be used on the n models running simultaneously. This would result in a maximum of n different values for the performance measure. One, of course, would still not know if the global optimum had been obtained. It is likely that new search techniques will be developed as the result of the availability of associative processors.

Summary

In this paper, we have attempted to point out some of the advantages of using the associative processor for discrete simulation. These include decreased running time, the ability to run several models simultaneously, and a more natural mapping between the problem structure and the machine structure. In order to realize these advantages considerable research is required in associative processor systems and their applications.

References

1. Bell, C. Grodon and Allan Newell. "Computer Structures: Readings and Examples", McGraw-Hill, 1971.

2. De Fiore, C.R., Stillman, N.J., and Berra, P.B. "Associative Techniques in the Solution of Data Management Problems", Proceedings-ACM National Conference, August 1971.

3. Dodd, G.C. "Elements of Data Management Systems" Computing Surveys, Vol. 1 No. 2, June 1969.

4. Emshoff, J.R. and R.L. Sission, Design and Use of Computer Simulation Models, The Macmillan Company, 1970.

5. Feng, T. "A Computer System Suitable for Criminal Identification", Proceedings of Carnahan Conference on Electronic Crime Countermeasures, April 1970.

6. Feng, T. "Personal Communication", June 1971.

7. Lewin, M.H. "Retrieval of Ordered Lists from a Content Addressed Memory" RCA Review, Vol. 23, June 1962.

8. Orlando, V.A. and Berra, P.B. "Associative Processors in the Solution of Network Problems", 39th National ORSA Meeting, Dallas, Texas, May 5-7, 1971.

9. Rosen, S. "Electronic Computers: A Historical Survey", Computing Surveys, Vol. 1, No. 1, March 1969.

10. Rudolph, J.A., Fulmer, L.C., and Meilander, W.C., "The Coming of Age of The Associative Processors", Electronics, February 15, 1971.

11. Slotnick, D.V. "Unconventional Systems", AFIPS, Proc. Spring Joint Computer Conference, 1967.

12. Stillman, N.J., De Fiore, C.R. and Berra, P.B. "Associative Processing of Line Drawings" Proceedings SJCC, May 1971.

13. Wilde, D.J. Optimum Seeking Methods, Prentice-Hall, 1964.

14. Wolinsky, A., "Principles and Applications of Associative Memories", Third Annual Symposium on the Interface of Computer Science and Statistics, Los Angeles, California, January 1969.

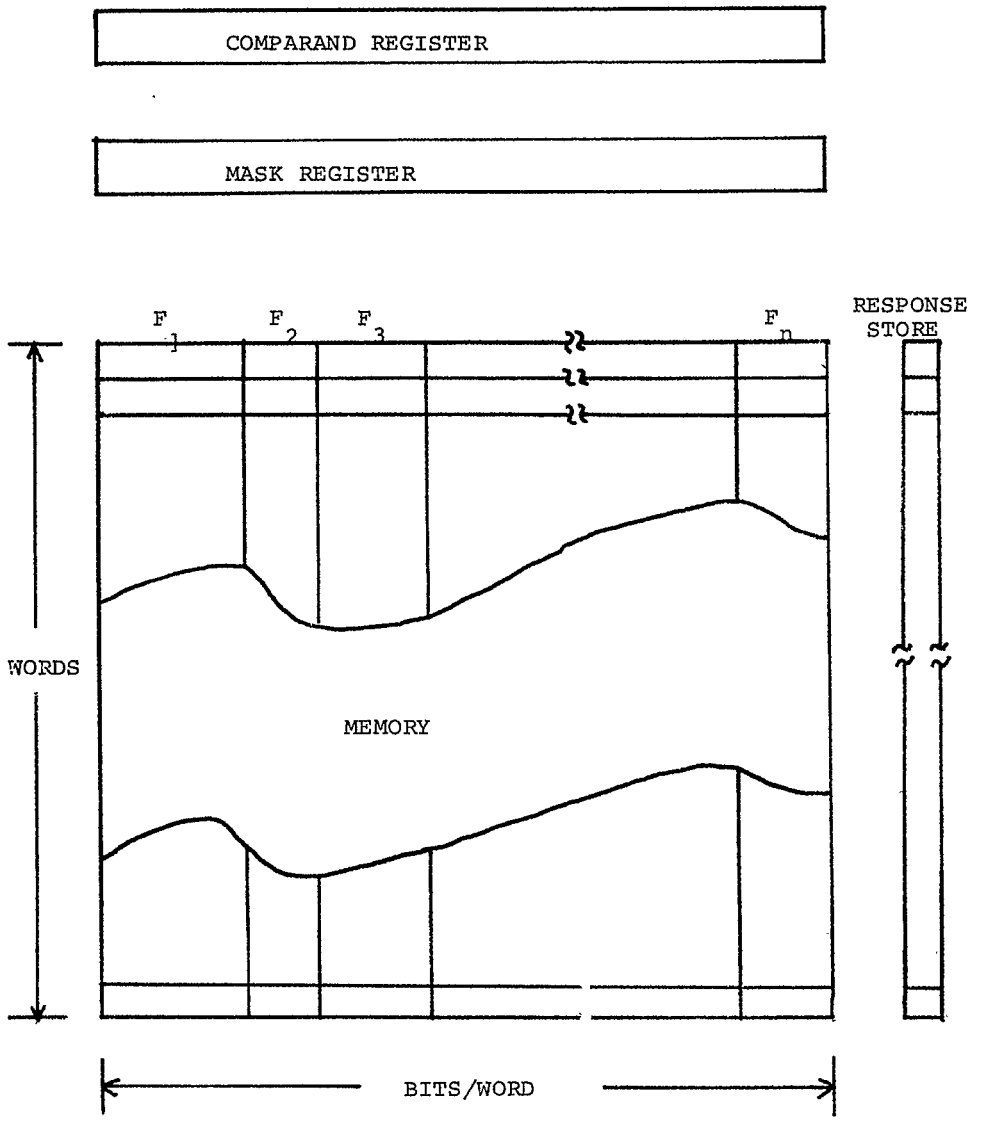


Figure 1. An Associative Memory Layout

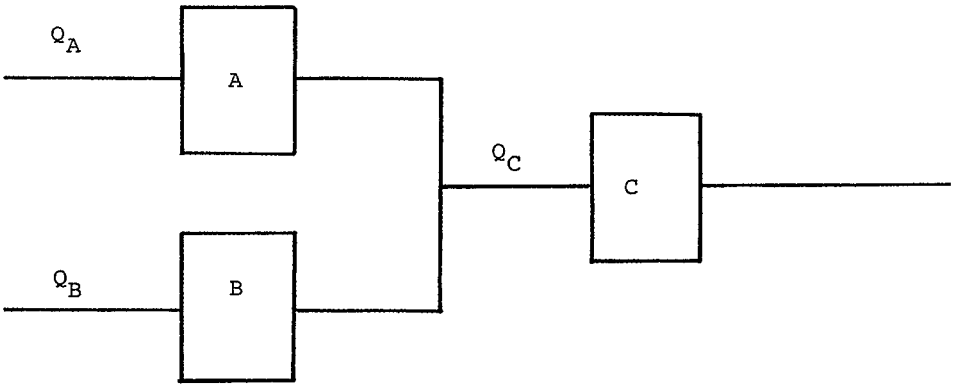
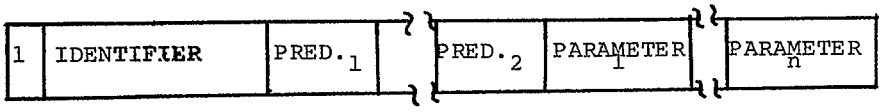
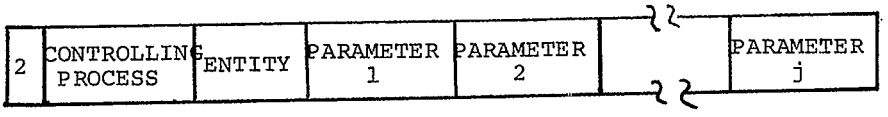


Figure 2. Simple Queueing Model



(a) Process Entry



(b) Queue Element Entry

Figure 3. Typical Data Entries