Program Design to Achieve Maximum Utilization in a Real-Time Computing System

A. FREDERICK ROSENÉ†

IN THE DESIGN of a real-time computing system, a major problem is the selection of machine capacity and speed adequate to handle the maximum data-processing rates and yet not burden the system at other times with unused machine capability. All considerations—cost, efficiency, size, good design practice—require maximum utilization of the computer’s complete capability. The design of the computer program to achieve maximum utilization is especially difficult in real-time systems which must be capable of handling a wide variation in data rate.

A real-time program design problem of this type was faced in the development of the Fire Control Center for the PLATO antimissile system. In addition to the requirement of handling a wide variation in traffic rate in real time, a Fire Control computing system must also be small and mobile. The program design which was developed illustrates one way in which a high-speed computer may be used efficiently for such a problem and is general enough to be applied to other real-time problems such as transportation or communications systems control.

The basic property of a real-time system is that certain commitments must be met at particular times no matter what the load on the system. The method of approach considered here uses as a design base a timing interval defined as the largest interval in which the essential outputs are required no more than once. The size of this interval, as a consequence of this definition, is determined by the external dynamic and physical environment, and is not dependent on the computer being used.

In addition to the essential set of outputs produced during a basic timing interval, there are also outputs, in most real-time systems, whose computation may be delayed to a later time. In the Fire Control application such tasks as predicting target position or selecting launch sites for a defensive missile could be delayed, whereas computing a command for a defensive missile which is airborne may not be delayed. In order to reduce computing time when an overload is present, two avenues of approach are available. (See Fig. 1.) First, the computation of the quantities which are not essential during the present interval can be delayed. However, before this is done, the expected load in the future, when the output of this computation can be delayed no longer, will have to be considered. The second alternative would be a change in the method by which the essential outputs are formed; that is, a sacrifice of quality for quantity. The high speed at which computers work makes it impossible for human intervention to control the selection of what is processed and the manner in which it is processed. Therefore, it is necessary for the computer to determine what it should do and how it should do it by examining not only present conditions but also predicted future conditions.

![Fig. 1—Methods for reducing computing time.](image)

A program for providing this type of control for a large system would soon become a logical monster offering little flexibility of operation and an almost impossible debugging job unless some type of division of responsibility can be obtained. The basic principle of design which was applied is a division of the system into sections, each of which has a control program associated with it. These sections are connected and controlled by a central control program which imposes a given strategy on the system by the manner in which it connects the sections of the system. One outstanding advantage of this type of program design is the flexibility inherent in the control programs as well as in the operational programs. Also, this design enables several sections of the system to be programmed concurrently by different programmers without the worry of incompatible results.

In the design of the PLATO Fire Control Center, consideration was first given to defining levels of decision which provide a basis for proper division of the system into sections. This was followed by a detailed design of the different sections and their associated data-storage design. Of course, during this design period full consideration was given to possible orderings of simulation experiments for the efficient analysis of the system.

**DIVISION OF THE SYSTEM BY DECISION LEVELS**

The system recognizes three levels of decisions: those appearing in central control, subcontrol, and system significant blocks. (See Fig. 2.) The basic unit of the

system is the system significant block in which all operational programs appear. Associated with each of these blocks is a subcontrol which controls its operations. The highest level, central control, controls the relationships between subcontrol-system significant block combinations.

The properties which a system significant block should satisfy are:

1) Any decision made in the system significant block must be a decision directly dependent on output within that system significant block, and must directly control the operation within that block.
2) Any functional change in the system significant block should not affect the rest of the system.
3) Any functional change in the rest of the system should not affect the system significant block.
4) Any section of the system significant block which satisfies the above three conditions must not be directly controlled by a timing decision (a decision determining how time is used).

Therefore, a system significant block becomes a black box provided with inputs which in turn generates outputs. The factor determining how the system is divided into system significant blocks is that no decision controlling operation of the rest of the system nor timing decision can be included within a block. Also, as the name implies, the system significant block should represent an operation which is meaningful to the system as a whole. For example, a Fire Control Center which must interpret data from a radar and, on the basis of these data, launch and guide a defensive missile to interception of an enemy missile, might be divided into systems significant blocks in the following manner. (See Fig. 3.)

1) A block which detects the presence of a target by examining the data received from the radar.
2) A block which smooths the target data and predicts the position of the target.
3) A block which predicts the position of the defensive missile.
4) A block which computes commands which are sent to the defensive missile.

In any computing system, the juggling of the ordering of decision can make it possible for almost any block to satisfy the properties of a system significant block. Thus, one would begin the division of a system by determining the operations present which actually possess significance to the system as a whole. Then the blocks performing these operations can be designed according to the four properties already listed.

Each system significant block will have associated with it a subcontrol block which controls its operation. A subcontrol block is composed of decision operations which are directly dependent on the output of a central control and/or one or more subcontrol blocks, and the result of these decision operations directly control the operation of one system significant block. This intermediate level of control associated with each system significant block provides a division of control into subsections in the same manner as the system significant block division divided the operational program into more manageable subsections. The subcontrol block has the responsibility for making all decisions, time-dependent or otherwise, which affect the operation of its system significant block, that is, affect the method in which the block is used. By way of illustration, the subcontrol of a system significant block which predicts future target position might be required to determine if it has been allotted enough time to predict the position of all targets which have been detected by the best mathematical method. When there is not enough time, the
positions of some targets will have to be predicted using a less accurate but shorter method; and the subcontrol will have to determine which targets should be processed using the shorter method.

The highest decision level is the central control which contains all decision operations directly concerning strategy. A decision at this level is directly dependent on the output of one or more subcontrol blocks and directly controls the operations of one or more subcontrol blocks. One important reason for separating decisions of this type from those appearing in the subcontrol blocks is that changes in strategy could be frequently required by a changing tactical situation.

UNIT DESIGN

The properties already discussed, which are associated with each of the three levels of the system, set the bounds for the detailed design of each level.

The structure of the system significant block, which is the basic unit of the system, is a collection of calling sequences. (See Fig. 4.) These calling sequences may be considered as macro instructions from which a program is formed for carrying out a designated task. The normal return of each macro instruction contains an unconditional transfer instruction; the addresses of these transfer instructions are inserted by the subcontrol block of the system significant block. This insertion of addresses, accompanied by the storing or changing of parameters in the macro instructions where needed, is the only operation necessary to form a program. Hence, the subcontrol block effectively writes a program by inserting transfer addresses and parameters in macro instruction. However, in order to accomplish this the subcontrol must determine on the basis of data supplied by central control what tasks are waiting to be done, in what order they should be done, and how they should be done.

Provision must also be made for an overload of work, that is, a selective process for determining what tasks should be delayed until a later time. In order to do this, information such as present work load, time available for the subcontrol and system significant block combination being used, priorities of tasks waiting to be done, and any restrictions placed on waiting tasks must be available. (See Fig. 5.)

The first function of the subcontrol will be to select a task. This selection will be predominately based on a

```
Fig. 4—Structure of a system significant block.
```

```
Fig. 5—Structure of a subcontrol block.
```

* Normal returns are transfer instructions whose addresses are supplied by subcontrol.

From the collection of the Computer History Museum (www.computerhistory.org)
comparison of priority to time available; however, work load could also be a factor. Once a task has been selected, it is necessary to determine how this task should be carried out. There are actually two types of selections which must be made at this point: a selection of method based on the same quantities on which selections of tasks were based, and a selection which is completely independent of timing requirements but based on the state of a given task. The first type of selection is concerned with quantity vs quality. In other words, as demands on time increase, accuracy may be sacrificed in order to increase the number of outputs computed during a computational interval. This type of decision must be based on parameters computed by the central control since it is concerned with the strategy being employed. The second type of decision is concerned with how much has already been done on a certain task and on the basis of past results what should be done next. During initial simulations there will be the need for the analyst to select arbitrarily a given method he wishes to evaluate. For this purpose a manual selection switch not under control of the subcontrol is included.

One final duty of the subcontrol is to place all inputs and outputs of the system significant block on tape in the binary mode if they are desired for historical records and use in future analysis of system operation.

The central control block is only entered at the beginning of each basic timing interval. (See Fig. 6.) At this time, the subcontrol-system significant combinations which are to be used, the order in which they should be used, and the amount of time allotted to each, is determined. In order to accomplish this, the central control must be able to assess the work load expected during this and succeeding intervals, and the relative importance of each task to be performed by the system significant blocks. This type of information would be based on such quantities as probability of a task obtaining its objective, the size of the waiting line for a given task, time required by each system significant block, and necessary data rates allotted to a given task. It would also be the duty of central control to interpret external commands which have been supplied since the last interval. This would include information regarding functioning of different sections of the system, changes in strategy, and instructions concerning the deletion of certain data.

Such things as error checks, system performance checks, and diagnostic programs will be handled by central control or at least controlled by central control. Signals indicating errors either in the computers or in other equipment such as data links will be interpreted by central control and the necessary action taken. This action would include such things as bypassing computations which cannot be handled with a given part of the system nonoperable, the switching-in of spare equipment, the calling-in of diagnostic programs to help locate the cause of an error, and the issuance of corrective instructions, if possible, where incorrect outputs have been given. System performance checks would indicate the way the system is operating and could be used to determine if an alternative strategy is required. Some examples of the types of decisions which central control may have to make are:

1) An external signal indicates that one launch site cannot be used. What action should be taken? The block which performs launch site selection would have to be notified; also, time must be allotted during the next few timing intervals for reassignment of launch sites for any interception already planned using the inoperative site.

2) There is a large number of targets requiring prediction and smoothing. How should the next time interval be used in order to reduce the size of this waiting line? During the next interval, as much time as possible would have to be allotted for the prediction system significant block. Hence, the other blocks would be allotted a minimum amount of time or bypass completely. The parameters sent to the prediction subcontrol would be such that all prediction would be done by the shortest method.

The exact structure of the central control cannot be specified in detail since the functions it must carry out are dependent on strategy. However, ideally, its function is to apply a strategy to the present conditions in order to produce a course of action which is consistent with immediate operational requirements and results in the least probability of system saturation.

---

**Fig. 6—Functions of central control.**

- **Representative Inputs**
  - Size of Waiting Lines
  - Data Rates
  - Error Alarms
  - Data from Control Center
  - System Performance Results

- **Representative Results**
  - Ordering and Time Allotments for SS Blocks During Next Interval
  - Parameter Changes
  - External Outputs to Monitors

---

From the collection of the Computer History Museum (www.computerhistory.org)
SUBROUTINE DESIGN

The macro instructions which compose system significant block have been described as calling sequences to subroutines. (See Fig. 7.) These subroutines are divided into three levels according to their use in the system, and hence the manner in which they must be written. The calling sequences of those in the top level only appear in system significant blocks and usually in only one block. The function performed by this type of subroutine is usually system dependent; that is, the function could be done in many ways and simulation and analysis will determine which method is best for a given mode of operation. Calling sequences of subroutines from lower levels could appear within a subroutine at this level. The calling sequences of a second level subroutine could appear in a system significant block or in a top level subroutine. The functions performed by a second level subroutine are explicitly defined mathematical or logical tasks and their use does not directly depend on mode of operation of the system. This level subroutine can only contain calling sequences of subroutines from the lowest level.

All subroutines in the top two levels are written in closed form and the location of all inputs and outputs are specified in the calling sequence. Provision is made within each subroutine for storing all inputs and outputs on tape under the control of its calling sequence so that any data needed for analysis purposes are readily obtained if desired. One of the differences between these two levels of subroutines is the way in which they are written; that is, the second level programs are completely general, whereas top level programs can be more specialized.

The third level of subroutines consists of utility programs, and the calling sequences of this type of program appear only within other subroutines of higher levels. All utility programs are written in a general closed form, and no provision for tape storage of inputs and outputs is provided.

Subroutines from all levels have error returns for overflow, division failure, and nonacceptable inputs.

DATA-STORAGE DESIGN

Another requirement of program design is the specification of the data storage. The data storage for a real-time system must be set up with the specific purposes of the system in mind and be compatible with the program design. Output requirements are one of the important considerations in designing data storage. System significant blocks and subroutines on the top two levels have provision for placing their inputs and outputs on tape. The data supplied by individual subroutines are necessary for analysis of the programs involved, whereas once the system is actually in operation, the output supplied by the subcontrol blocks is of prime importance. Since, in many cases, there is a minimum of time available for output in an operating system, the data-storage design should be such that the system significant block outputs require a minimum amount of computational time.

Even after a system is operating, it may be necessary to change or modify the strategy being employed. In order to expedite this type of change, separate storage for control data is advisable. In systems where random-access memory is limited, data not being used would be transferred to some auxiliary type storage such as drums. The data storage in this case would have to be designed with this transference of data in mind.

SIMULATION EXPERIMENTS

The first units of the system to be simulated are the subroutines. (See Fig. 8.) The purpose of this phase of simulation is to evaluate the mathematical models used

<table>
<thead>
<tr>
<th>Level</th>
<th>Use</th>
<th>Form</th>
<th>Location of Calling Sequences</th>
<th>Subroutines Used by a Subroutine</th>
<th>Tape Output Provided</th>
<th>Error Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dependent on Mode of System Operation</td>
<td>Specialized</td>
<td>SS</td>
<td>Levels 2 or 3</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Not Dependent on Mode of System Operation</td>
<td>General</td>
<td>SS or Level 1</td>
<td>Level 3</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Utility Programs</td>
<td>General</td>
<td>Levels 1 or 2</td>
<td>None</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Fig. 7—Subroutine levels.

| Phase I | (a) Subroutine.  
|         | (b) Groups of subroutines.  
| Phase II| (a) System significant block with all or part of its subcontrol.  
|         | (b) Groups of system significant blocks each with its subcontrol and part of central control.  
| Phase III| Complete system including peripheral equipment.  

Fig. 8—Ordering of simulation experiments.
over the range of inputs expected. In some cases, this simulation consists of connecting several subroutines together. The second phase consists of system significant block simulation accompanied by subcontrol control. During this phase, the different methods available for performing a function are assessed and the optimum selected. More than one method per function is kept in some cases because of time considerations, e.g., time available vs accuracy needed. Sections of the control program used during this phase become part of the subcontrol of the blocks being simulated. The third phase, the entire simulation, is reached by successively combining system significant blocks and adding the appropriate sections of the central control. Adequate evaluation of programs at any level is impractical without realistic inputs which are most easily supplied by other subcontrol system significant block combinations and central control; thus complete system simulation seems necessary. Also, a synthetic environment would have to be provided by peripheral subsystem simulation of such things as radar and missile.

Conclusion

For the PLATO antimissle system Fire Control Center, a program design was developed which embodies three decision levels: central control, subcontrol, and system significant block. This versatile program design provides for the maximum machine utilization, and permits a trade-off between quality and quantity of computation as well as expediting programming, debugging, and simulation.

The general concepts of this program-design approach should be applicable to other real-time computing systems in the business and scientific fields.

Pattern and Character Recognition Systems—Picture Processing by Nets of Neuron-Like Elements

L. A. KAMENTSKY†

INTRODUCTION

Spatial pattern recognition, of which the recognition of alpha numeric characters is a subclass, is an important and practical problem. More efficient coding of transmitted pictorial information and more efficient utilization of humanly produced information could result from its solution. The problem of pattern recognition has been stated as the assignment of a meaningful code to a recognizable structure in a set of signals. The signals, in this case organized spatially, are the result of a transformation from a visual picture field \( P \) to an electrical representation of this field. The points of this signal field \( S \) correspond to a characteristic of points in the picture. In this study, the reflectivity of given picture-point areas is quantized as black or white as a basis for a two-state electrical-signal representation of points of a pattern.

The total information content of this signal field is certainly less than that of the original picture. Useful pattern recognition requires that the code assigned to a pattern have even less information content than the signal field. For example, a code assigned to a pattern of a number in the set of 10-decimal-number symbols indicates the number value but need not reflect the size, position, nor quality of the original pattern. The pattern-recognition machine is thus a device for performing an information-destructive transformation on the signal field to yield an output code assigned to the value of the pattern. Internally, the machine may perform a sequence of information-destructive transformations. Furthermore, the machine may internally produce transformations yielding parameters of the picture as an intermediate step, rather than the output code directly. Such parameters may have a physical meaning. For example, in useful number recognition the presence of the pattern parameters, straight lines, openings in curves lines, or corners is significant for generating the output code.

This paper describes an approach to the solution of pattern recognition that may be characterized as "spatial operations by neuron-like elements." Signal fields are transformed by a predetermined network of threshold-responsive elements. These elements have been called neuron-like in that they have many inputs and a single all-or-nothing output; they are connected in spatial arrays with excitation or inhibition gating between the signal field and the elements, or between the output and inputs of elements. More correctly, they should be called spatial neurons, since only their spatial properties approach the assumed properties of neurons.

† Bell Telephone Labs., Inc., Murray Hill, N. J.