9-1-1967

Problems Encountered With Control Networks in Highly-Restructurable Digital Systems

Donald F. Wann

Robert A. Ellis

Mishell J. Stucki

Robert M. Keller
Harvey Mudd College

Recommended Citation

Wann, D.F., R.A. Ellis, M.J. Stucki, and R.M. Keller. "Problems encountered with control networks in highly-restructurable digital systems." Digest of the First Annual IEEE Computer Conference (September 1967): 29-32.

PROBLEMS ENCOUNTERED WITH CONTROL NETWORKS IN HIGHLY RESTRUCTURABLE DIGITAL SYSTEMS

D.F. Wann R.A. Ellis M.J. Stucki R.M. Keller
Washington University
St. Louis, Missouri

ABSTRACT. This paper discusses problems encountered with control networks in highly restructurable digital systems. In particular the treatment of implementation errors is covered with emphasis on concurrent processing. The implementation of concurrent processing networks may result in errors which will be quite complex to detect and systematic methods are warranted. Four meta control elements are employed in obtaining convenient concurrent structures. We analyze several error detecting schemes and conclude that the arc-node method with node partitioning appears to be the most realistic approach at this time.

Introduction

In this paper we discuss aspects of concurrent control structures encountered in the context of the Macromodular Project being carried out at Washington University. (1)(2) Although many of the ideas to be presented apply specifically to macromodular systems, we feel that these systems are the first of a new class of highly-restructurable systems, (3)(4) and that the considerations to be discussed are characteristics of this class.

Macromodules are digital building blocks that are rather unique in that data processing structures constructed from them are isomorphic to a directed graph and can, therefore, be designed and analyzed from a graph-theoretic viewpoint. The nodes of the graph correspond to the macromodules themselves and the arcs of the graph correspond to the control cables that interconnect the modules. All electrical and mechanical considerations have been eliminated by careful design of the units, and the design of data processing structures is therefore an exercise in logic only.

The macromodular concept makes possible data processing avenues not possible in fixed structured systems. Specifically, the ease with which macromodular structures may be re-configured and augmented makes it possible for a system user to alter a system's configuration so as to be more efficient for his particular problem area. There are many ways in which the efficiency of a system may be affected: machine instructions may be changed to better fit the problem area, machine instructions may be re-designed in a highly parallel or concurrent fashion to minimize the operating time of the system, etc.. As is to be expected, however, opportunities such as these are not without accompanying pitfalls. It has been discovered that in the process of formatting a system alteration and implementing it into the system it is extremely easy to make what we shall call "implementation errors" as contrasted to algorithm errors. The latter imply an incorrect concept of the problem and indicate an error on the part of the person developing the mathematical model. Implementation errors, on the other hand, refer to errors encountered when encoding an error free algorithm into a directed graph. A directed graph is composed of many types of nodes, each of which has specific functions and specific rules of usage. As a system becomes larger or more complex, it becomes increasingly difficult for the designer to visualize the system in its entirety and it becomes, therefore, increasingly probable that some node's rule of usage will be violated. If a system is designed or reconfigured according to a graph containing an implementation error, the system will malfunction. This is not a desirable way of detecting errors, however, since the amount of time spent in locating and correcting the source or sources of malfunction may be greater than the time advantage to be gained from the restructuring. If, therefore, a highly-restructurable system is to be used to fullest advantage, debugging time must be minimized and, if possible, made automatic.

The problem of automatic detection of implementation errors is the authors' area of interest and the topic of this paper. The approach presented is to represent the directed graph as an arc-node connection matrix and then by a systematic reduction procedure, attempt to reduce the matrix down to a single cell. If this is possible, the graph was error free; if not, an error exists at that step in the procedure where further reduction is impossible.

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense through contract SD-302 and by the Division of Research Facilities and Resources of the National Institutes of Health through grant FR-00218.

Control Elements

When considering concurrent processing, techniques for the manipulation of data remain essentially the same as in standard sequential processing, with perhaps the need for some type of interlock to mediate conflicting concurrent requests for data, as from memory. Discussion of these considerations can be found in Bernstein's work. (5)

In considering control operations, however, the need arises for new types of control devices, for the elementary control transfers of sequential programming are not sufficient to accomodate the wider variety of operations encountered in concurrent operations. In constructing asynchronous systems utilization of distributed control instead of the conventional centralized control is advantageous. This distributed control must be manipulated, or changed, and this is performed by means of "control elements". The general control element is depicted in Figure 1 in which {I} and {E}

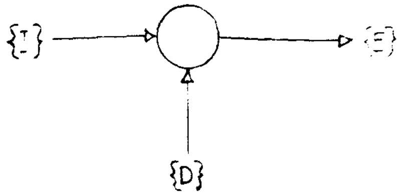

Figure 1. Generalized Control Element

are sets of incident and extant control lines and $\{D\}$ is a set of data. Thus a typical output line E_i can be described as a function of the incident control and the incident data, that is, $E_i = f(\{I\}, \{D\})$. Although numerous control devices are conceivable, we shall define but four for use in the present analysis. These four meta control elements are the branch, rendezvous, decision, and merge. These are denoted by B, R, D and M respectively as shown in Figure 2. The branch

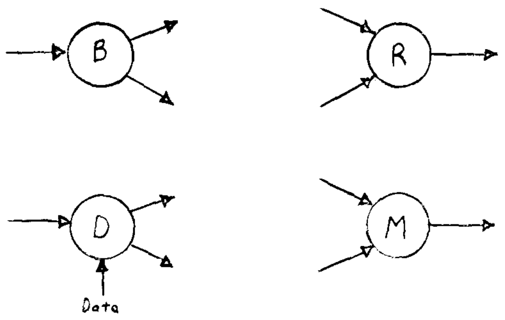

Figure 2. Meta Control Elements

produces two simultaneous extant control signals which are independent of incident data. Thus the branch allows the initiation of parallel operations. The rendezvous element (whose operation is also independent of data) terminates concurrent sequences by yielding an output only after both incident signals have been received. The decision element produces an output on only one of the extant lines, the particular line being dictated by the value of the incident data. The merge element produces an output for a signal on either of its incident lines; its operation is not defined if its input signals are not mutually exclusive. Similar control functions (such as DO TOGETHER, WAIT, FORK, JOIN, etc.) have been described in the literature pertaining to higher level languages for multiprocessing (6) but we are referring here to actual hardware devices rather than program statements. To illustrate both the use of these elements and also to show how implementation errors may easily intrude into a program, consider the computer statements:

Statement Number	Statement
	Transfer contents of register B to reg. C
	Complement contents of register A
1	Add one to contents of register A
	If A = 0 then go to EXIT else go to 1
Exit	....

This may be represented in abbreviated form as:

```
B ⟶ C
COMP A
INDX A
A = 0?, 1, EXIT
```

One implementation of this algorithm in a concurrent flow-graph is shown in Figure 3. Observe that this implementation of the algorithm contains an error, for when a NO decision is made, the recursive path, PQS will be stopped at the rendezvous element -- permanently -- and the program will not function properly. This then is an implementation error. Certainly the detection and correction of such errors is quite simple in such an elementary configuration; two possible solutions are shown in Figures 4 and 5; but, in more complex algorithms this detection becomes exceptionally difficult and automatic error detection schemes most certainly are warranted.

Analysis of Control Systems

The problem of analyzing a design in which decision-making and parallel processing are highly interdependent has several solutions, such as exhaustive testing, simulation, logical equation manipulation and list processing. However, a graph theoretic viewpoint, in which manipulative techniques on an arc-node connection matrix are used, is emphasized in this paper. A method for partitioning a graph into classes of compatible nodes is developed in which a class of compatible nodes is defined as subgraph having only one incident and extant arc to the class, and all other arcs attached to nodes in the class are either incident or extant to nodes within the class.

Using the partitioning technique and several theorems relating to simple configurations of nodes, this technique can reduce practically all realistic designs to a few small complex subgraphs. At this point in the analysis two alternatives are postulated. First, these complex subgraphs could merely be presented to the user for verification. Second, these subgraphs could be subjected to detailed automatic analysis techniques. (Which are too time-consuming if the entire flow-graph is analyzed.) To illustrate how these techniques are applied to more complex algorithms several examples are presented in this paper. The control structure for one of these, a floating point arithmetic unit, is pictured in Figure 6.

Conclusions

In this paper we have introduced a new degree of freedom available to the computer user-programmer when employing a restructurable computer for implementation of concurrent processing. Discussion of errors arising in this context have been analyzed and several error detecting schemes presented. The arc-node method with prior node partitioning appears to be the most realistic approach at this time. In the future we believe that increasing importance will be placed on these aspects of error detecting and the systhesis of concurrent processing algorithms. However, even until more sophisticated solutions are found, many important and useful systems may be constructed using the elements and methods we have discussed.

References

1. Ornstein, S.M., Stucki, M.J., and Clark, W.A., "A Functional Description of Macromodules", Proc. S.J.C.C., pp. 337-355, 1967.
2. Stucki, M.J., Ornstein, S.M., and Clark, W.A., "Logical Design of Macromodules", Proc. S.J.C.C., pp. 357-364, 1967.
3. Clark, W.A., Stucki, M.J., and Ornstein, S.M., "A Macromodular Approach to Computer Design", Technical Report #1, February, 1966, Computer

Research Laboratory, Washington University, St. Louis, Missouri.
4. Estrin, G., "Organization of Computer Systems: The Fixed Plus Variable Structure Computer", Proc. W.J.C.C., pp. 33-40, 1960.
5. Bernstein, A.J., "Analysis of Programs for Parallel Processing", IEEE Transactions on Electrical Computers, Vol. EC-15, pp. 757-764, October, 1966.
6. Dennis, J.B., and Van Horn, E.C., "Programming Semantics for Multiprogrammed Computations", Comm. of the ACM, Vol. 9, pp. 143-155, March, 1966.

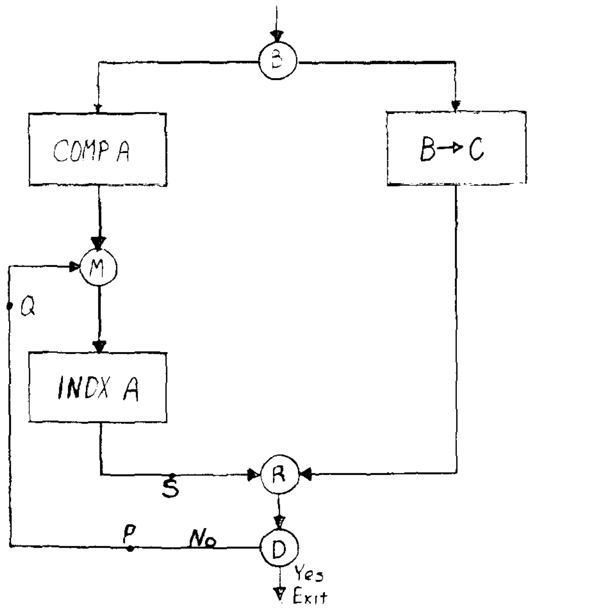

Figure 3. Incorrect Implementation of Algorithm

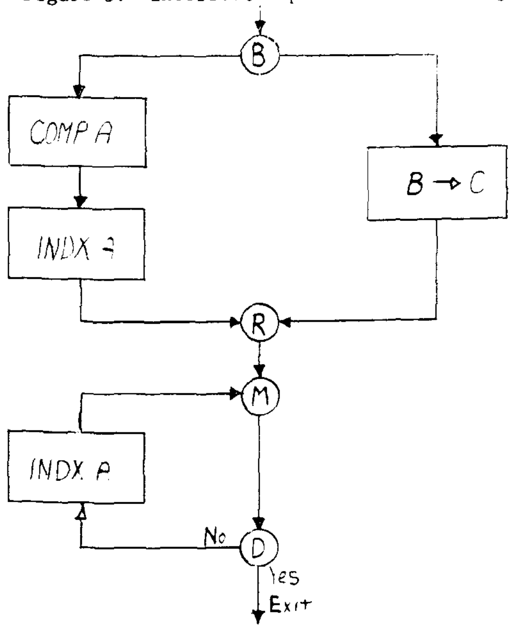

Figure 4. Correct Implementation - Case A

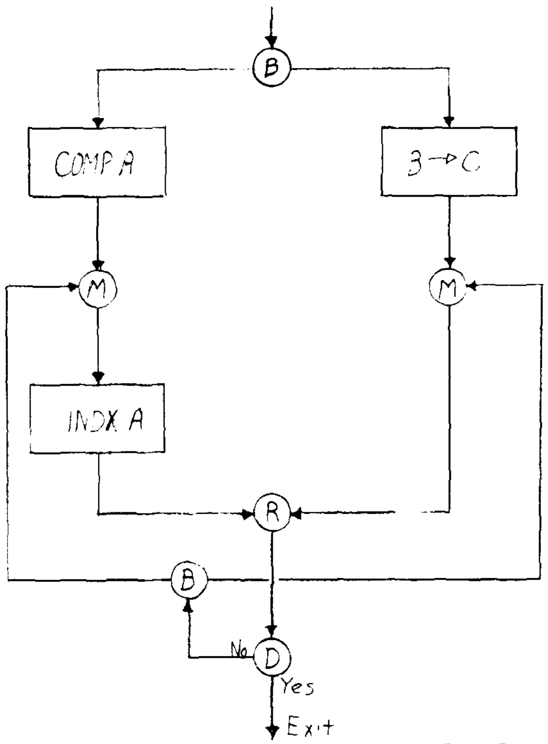

Figure 5. Correct Implementation - Case B

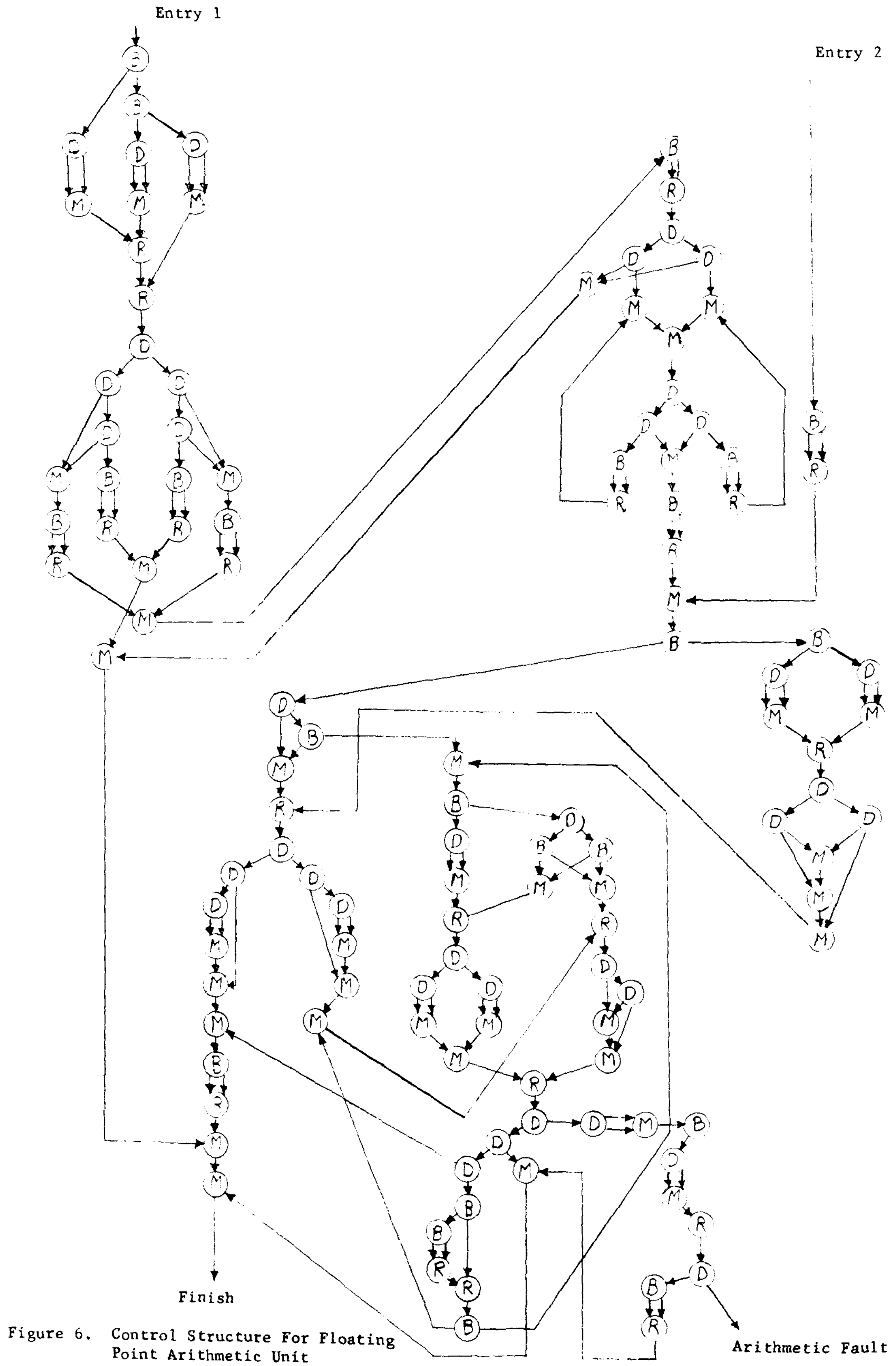

Figure 6. Control Structure For Floating Point Arithmetic Unit