Reviews of Books and Papers in the Computer Field

DONALD L. EPLEY, Reviews Editor
D. W. FIFE, A. J. NICHOLS, A. I. RUBIN, H. S. STONE
Assistant Reviews Editors

Please address your comments and suggestions to the Reviews Editor: Donald L. Epley,
Department of Electrical Engineering, University of Iowa, Iowa City, Iowa 52240

A. SYSTEM DESIGN


With this paper the authors are announcing development in progress on a simulation programming language which they have named SODAS (Structure Oriented Description and Simulation). Motivation for the development is provided by an opening discussion of desirable attributes of a language for general system design. A brief description of SODAS syntax and semantics follows, and then a SODAS program for simulation of a highway traffic control system, a fairly detailed example of how the language fulfills the desired objectives.

SODAS is designed for simulating discrete-time systems, where changes in the system state may only occur at discrete “clock” times. Systems with asynchronously operating components may still be simulated, but this means that the description of each asynchronous component must keep track of the simulated time. Components may be inactive for designated periods of time or until some Boolean condition on inputs becomes true. SODAS provides for simultaneous events occurring in a system, and for components which have an instantaneous response.

The example program shows that SODAS will incorporate a number of sublanguages, so that the SODAS user may choose from them the language best suited for the description of each system component. The different languages give flexibility in the nature of components and in the level of design detail. Among those illustrated are SFD-ALGOL, BOOLE, and SIMULA, although none of them are discussed in this paper, since with the exception of BOOLE, they have been the object of previous papers.

Two further points about SODAS are worth mentioning here. The language is “recursive” inasmuch as a “system” description (i.e., program) may be used without modification as a “component” description in a larger system. This is not especially illustrated by the example program, which rather focuses on the point that the language allows a mix of structural and behavioral description. In the example, the design of the traffic control unit is approached in three steps, showing increasing detail in the hardware structure determined for the unit. At the first step, the control unit is a “black box” and its description is therefore purely behavioral. On the second step, specific parts of the control unit are identified and described as black boxes, thereby giving structure to the unit. On the third step, the internal structure of several components is described at a digital logic level using the BOOLE language.

SODAS appears to be an important step in the development of system design and simulation aids. This paper is sufficient to whet the reader’s curiosity, but is not adequate as a tutorial or even an introduction. The discussion and description are scanty and poorly organized in places and the reader is occasionally left “dangling.” For example, the authors mention that the preliminary implementation of SODAS now in progress relies on a “slightly extended” form of precedence analysis, the latter having been developed by Wirth and Weber in their paper on EULER.1 But nothing further is said about the nature of the slight extension or why it was needed or desired.

DENNIS W. FIFE
Lexington, Mass.


Second generation computing systems rarely had an aggregate of peripheral devices with high enough data transfer rates to tax memory speed or to degrade central processor (CP) performance. Whenever they did, there was no attempt to overlap transfer with computing, and the CP remained idle. But in many third generation systems, there are several processors—both I/O devices and arithmetic units—operating asynchronously and independently, each of which has a high transfer rate to main memory. Thus, the question of designing sufficient bandwidth into memories and busses is significant. Experience with some of the commercial multiprocessor systems (e.g., UNIVAC 1108 or IBM 360/67) shows, however, that sufficient bandwidth is not enough. The system must be organized so that the full bandwidth, or at least a large fraction of it, can be used effectively. It is this problem to which the author has ably addressed his paper.

The problem is briefly the following. In the computer system, there are two kinds of devices which require memory service: the fully buffered and the unbuffered. The unbuffered devices are those such as disk or communication lines. When one of them is transferring data, an unusual delay in the response of the memory module can cause “overrun,” a condition where information is lost or clobbered by following data. When it occurs, the transfer operation must be restarted from the beginning at relatively expensive cost. The fully-buffered devices, such as a CP or a buffered printer, do not suffer this restriction. They are capable of waiting indefinitely for service by a memory module, and the only cost is that of the delay itself.

Most systems avoid overrun by assigning priorities to the various requests from the devices to memory. The memory bus which serves the fastest devices is typically the highest priority, and buffered devices are the lowest. Whenever a drum and a CP simultaneously request memory service, the CP is forced to wait one cycle while the drum is served. The resulting degradation is a function of the probability that simultaneous requests can occur. This reviewer has independently measured that value for one system, and it was discovered that the CP operates at 38 percent of its normal speed when the complement of I/O devices is transferring data. This situation is the