Block Diagrams in Logic Design

LOWELL S. BENSKY
NONMEMBER AIEE

THE THESIS proposed in this paper is: While logic equations and even Boolean algebra are not essential to the block diagram approach to logic design, considerable penetration of block diagrams into the logic equation approach is unavoidable. In support of this thesis, several aspects of logic design will be investigated in the body of this paper. Emphasis will be given to the creative nature of logic design.

The generation of basically new configurations and new logic systems will be discussed. Next, the question of accommodating nonlogical elements will be explored. Finally, some problems in the transition from logic design to physical equipment will be pointed out.

Logic design, as the term will be defined and used here, is in the last analysis an art. The individual logic designer, like the artist, must develop his techniques according to personal preference, training, and ability. A discussion on the basic approach and execution of details for a logic design must consider the individual and the specific problem; no universally optimum logic design procedure, block diagrams or logic equations, will be found.

Definition of Logic Design

For the purposes of this paper logic design will be taken to include the activities pursued by logic designers in conjunction with system analysts, programmers, circuit and equipment engineers, and operations-maintenance planners resulting in the detailed interconnection and synchronization of a set of logical elements which satisfy the requirements for a digital computing device. Working with system analysts, the logic designer will determine from the problem specification: computation rate, input-output rates, input-output data format, capacity and types of data storage, and word length. With the help of programmers, the logic designer will determine: word length, operations required, instruction execution times, and programming aids. Together with circuit and equipment engineers, the logic designer will specify physical and operating characteristics such as: volume, weight, power, reliability, acceptable environmental conditions, and available hardware techniques. The logic designer and operations-maintenance planners describe the level of field programming and operating and maintenance personnel.

Based on these specifications and requirements, the logic designer will take one or more cuts at the gross logic of the computing device. Sample programs, cost estimates, and manufacturing schedules are worked out for each cut. The most likely candidate will be logic designed in detail and checked thoroughly. Exhaustive programming will be performed to insure adequate problem satisfaction; logic changes are introduced as required. Working closely with circuit and equipment engineers, the logic is modified to accommodate circuit restrictions and to exploit hardware techniques. Then the logic is put in form convenient for establishing wiring diagrams. After the computing device is fabricated, the logic designer is a member of the debugging and checkout team. Errors and unwanted redundancies which show up in this phase are corrected by the logic designer. Finally, the logic designer plays a major role in preparing preventive and corrective maintenance procedures. All of these activities are essential parts of logic design. The arguments which will be presented have significance when interpreted through this definition of logic design.

Creative Aspects

To derive a logic design to solve a new computing problem or to substitute a more elegant design for a brute force approach requires a measure of creativity. An important ingredient in the creative process is unencumbered visualization. In the case of logic design, creative efforts depend on incisive visualization of both spatial arrangements, timing and the interaction between them. When the logic designer is faced with the problem of forced invention to solve a problem or when he attempts to invent in order to improve on an immediately obvious or known solution, the inventive process is facilitated by a complete comprehension of: 1. data flow paths, 2. the factors making up control information, 3. the timing relationships among computer units, and 4. the interdependence among these items. Further, since new ideas are very often fleeting phantoms, comprehension which measures practicality should be rapidly attainable. A block diagram of a computing device affords rapid and complete comprehension of: 1. the required major registers, counters, and memories, 2. the data transfer paths and associated control gates, 3. the role of control information, and 4. the required timing and synchronization. Creativity and invention in taking initial cuts on the gross logic of a machine or in detailing the final logic are enhanced by the over-all perspective characteristic of the block diagram approach. The logic designer using the logic equation approach is likely to bog down in symbolic representation and abstract equations diluting the opportunities to invent.

Since the early days of the Electronic Discrete Standard Analog Computer (Edsac), Harvard, Electronic Discrete Standard Analog Computer (Edvac), and Impulse Automatic System (IAS) computers, several basically new logic designs and new logic ideas have been created. Solidifying and testing these new ideas were facilitated by use of block diagrams.

Block diagrams paved the way for new logic ideas by showing functional feasibility and/or economic advantage: for integrating a new part into an existing structure, for replacing a part of a structure with a different implementation, or for assigning additional tasks to existing parts of the structure. It might be interesting to speculate on how many of these developments would not have emerged had logic designers depended exclusively on logic equations, restricting block diagrams to the simplified picture. The B-box, or instruction address modifier, would probably not have joined the ranks of such standard computer elements as the instruction counter.

The pseudo-parallel arithmetic using a phased clock for high-speed computation would not have been invented and used in computers like the Livermore Atomic Research Computer (LARC) and UDOFT. Variable word length and several of its important implications would not have been applied to digital computers like the International Business Machines Corporation (IBM) 705 and Radio Corporation of America BIZMAC.

For example, the freedom to rearrange the variable length items of a BIZMAC message during read-in from magnetic tape to high-speed storage (random composition on input) would not have been recognized as a feasible addition to the

BIZMAC order code. This operation allows the programmer to specify an arbitrary arrangement of the items of a message in the high-speed store, which arrangement may be different from the sequence of the items on the input tape. This rearrangement is accomplished with items of variable length even though each item may not be consistently present in a group of messages. In this example, several new ideas were generated during the gross and detailed logic design phases: 1. Integrating into a 3-address computer a variable length instruction having many more than three addresses; 2. Releasing memory locations holding this instruction and making them available for input message storage; 3. Changing the mode of operation under control of input message characteristics and allowing flexible cycling among the several modes. These ideas were to a large extent fostered by the complete comprehension of a block diagram showing previously established flow paths, timing, registers, and control flip-flops. The latent possibility of random composition on input would have been lost if this instruction had been designed by adding terms to the set and reset equations of flip-flops established for the other instructions and for a straightforward version of the read-in instruction.

Nonlogical Elements

The term "nonlogical element" might imply that in logic design these elements assume secondary importance. Yet, in practice nonlogical elements, which solve circuit restriction problems, marriage problems between separate units, and timing problems, are essential and often represent a major fraction of the equipment in a computing device. It is, therefore, important that these elements be introduced during the logic design process and not as an afterthought following the completion of the design. Only in this way can an over-all equipment optimization be approached, rather than just an optimum logic design.

A discussion on the accommodation of nonlogical elements can be divided into two parts: internal logic design and input-output device integration. Internally, digital computing devices require cathode or emitter followers, clamping circuits, pulse amplifiers, nonlogical delays, pulse stretchers, etc. Such nonlogical elements have a direct effect on: the number of components in a machine, the size and weight, power dissipated, cost, and operation reliability. Further, nonlogical elements have an effect on logic design minimization decisions. For example, one may choose central decoding of a function and use power distribution of the decoded levels or pulses to their points of application; or it may be more advantageous to distribute the function and decode separately at each point of application. A similar decision may be encountered between the use of separate corrective timing delays or a more complex time pulse distributor. Use of block diagrams will rapidly yield the number and physical location of inputs to be driven by each flip-flop, decoder, and pulse source output. Decisions on constraints or modifications to the logic can be made directly. Nonlogical elements can then be included on the diagram using simple and unobtrusive symbols, thus insuring that the elements are not overlooked.

Similar arguments apply to pulse standardizers, timing oscillators, one-shot multivibrators, and terminating or coupling filters as used in the marriage of input output devices and the central computing device. Decisions on the types of input output devices, their modes of operation, and associated modification and addition to the computer logic are facilitated by block diagrams. The use of block diagrams permits the evaluation of several eligible input-output devices. A decision to select a given device can be made with proper consideration to associated nonlogical elements. For example, the opportunity is minimized to neglect special matching filters required when a telephone line is used as an input to a data processor. Again, a block diagram will minimize the chances of deciding that a computer shall drive a group of tabular and graphical displays, forgetting to include the bank of buffer amplifiers required.

Space Allocation

The next argument is weak for computer development and construction programs which are allowed to proceed leisurely. Since this is usually not the case, the questions of what kind and how much rack, cabinet, or panel space cannot wait on a final logic design. A reasonable estimate can be made on the quantity of space required for each unit in a computer from a block diagram in process. This estimate can be used in selecting one of several logic approaches and it can also be used to give the mechanical and fabrication activities much needed lead time.

Another point in this connection is the use of the block diagram to come up with a physical configuration. The number of leads passing from unit to unit in the computer and special conditions imposed, such as lead length, separation from chassis and other leads, etc., are quite evident from a block diagram showing logical and nonlogical elements. These factors can then be used to locate each separate unit within the physical computer package.

Conclusions

It is the author's belief that the block diagram is the sine qua non for creative, efficient, and complete logic design. It is required not only in the conceptual phase, where several gross logic designs must be evaluated and creativity must be fostered, but also in the detailed logic design, in coordinating with circuit engineers, and in planning a construction program. Concession to the opposition is appropriate when a large, relatively pedestrian computer is called for and if: 1. the detailed logic design is a significant fraction of the total computer cost, and 2. the techniques and machine time are available for going directly from equation to wiring schedule with complete checking on a computer.

Except on these conditions, the author recommends employing a well-ordered block diagram and avoiding a more academic approach.
Logical Design Methods

R. K. RICHARDS
ASSOCIATE MEMBER AIEE

IT WAS recently reported in a magazine that Theodore Roosevelt, when a student at Harvard, objected to the idea of arbitrarily assigning a student to a topic to be defended in a debate regardless of the beliefs and convictions of the student. He felt that this practice was not in the best interests of sincerity and intensity of conviction. In the present instance the author has been asked to defend the use of block diagrams in computer design in contrast to the use of logical equations (Boolean algebra). Inasmuch as the author has made frequent use of logical equations, it would seem appropriate to, first of all, make a statement as to my true thoughts on the matter to allow this panel discussion to be maintained on a basis which is as sincere and objective as possible.

As the chairman, Dr. Amdahl, has already pointed out, for purposes of this discussion the use of block diagrams does not necessarily imply that Boolean algebra is not to be employed at all. It may still be used as an aid to design and as an aid to notation in portions of a computer, but it is implied that the main body of the engineering design, production, and maintenance effort is to be accomplished through the medium of block diagrams rather than logical equations. The question is whether or not it is advisable to go all the way and eliminate block diagrams completely insofar as the basic logical functions are concerned. Instead, complete reliance would be given to logical equations for designing the computer, for communicating the design to the manufacturing department, and for leading the maintenance engineer to potential and actual defects in the machine. The author’s true thoughts, then, are that Boolean algebra is a very important tool for computer applications but that the advisability of attempting to rely on it entirely is open to question. As is discussed in more detail later, some types of computer components and circuits are such that it is substantially impossible, with the present state of the art, to get very far with an algebra in a manner that is useful. With other types of components and circuits it is quite possible to represent the entire computer by means of logical equations, and this has been done in some cases. Therefore, in a sense, a discussion of the relative merits of block diagrams and logical equations reduces to a discussion of the relative merits of the different physical realizations of logical functions. In addition to this factor in the comparison, there are other important factors, to be discussed later, which would tend to make it desirable to retain block diagrams even in instances where the circuits and components are of a type that allow a complete shift to Boolean algebra.

The first point is more of a warning than a concrete reason why block diagrams are superior to logical equations. Scientists, mathematicians in particular, are prone to use the concept of “elegance” in judging the merits of a solution of a problem. Many problems have more than one solution, with some solutions being more elegant than others. Individuals strive to find solutions which are as elegant as possible because such solutions command more prestige and a higher status in the scientific community. Elegance is a rather difficult property to define, but it seems that by almost any set of standards logical equations are much more elegant than block diagrams. However, one must be extremely careful in drawing conclusions from this fact because intellectual elegance is often a poor measure for weighing the merits of a proposal when something as down-to-earth and practical as a computer is involved. A good illustration of this point is in the very use of computers to solve mathematical problems. It has long been known that many types of mathematical problems, most forms of differential equations for example, can be solved readily by numerical techniques even when they do not yield to any known analytical approaches. However, numerical solutions have never been considered elegant; consequently, many mathematicians have shunned numerical approaches to problems in preference to the more elegant analytical approaches in spite of their limited application and usefulness. Now that computers are becoming so expensive and glamorous the interest in numerical solutions is increasing rapidly. It might be argued that in colleges a course in numerical analysis is now even more important than a conventional course in differential equations, but that is getting off the subject. The purpose here is to emphasize that the most elegant approach is not necessarily the most desirable approach, and promised advantages of an elegant approach are not necessarily always realized when the problem is set in a practical environment.

As is now well known, there are many different ways to go about the design of a digital computer. Even a casual inspection of a representative sample of the computers as built by different industrial, educational, and government organizations will reveal a vast difference in the nature of the components and circuits or in what is sometimes called the “design philosophy.” It has often been noted that with each different set of components and circuits there are a relatively few basic logical functions being performed. These functions are “and,” “or,” “not,” “delay,” and “storage,” or certain combinations of these functions such as “exclusive or” and “inhibit.” However, in spite of the common denominator which the basic functions afford, it is not true that all of the different approaches are equally amenable to Boolean algebra.

When adapting Boolean algebra to a computer in its entirety, it appears to be the situation that the most success is achieved when the following concept is employed in both the logical design and the circuit design. The computer is viewed as consisting of a large number of bistable elements, and time is divided into a series of equal increments or steps. At the end of each step, the binary values stored in the bistable elements are functions of the binary values at the termination of the previous time step. The relationship between the binary values from one step to the next can be represented by logical equations. A major limitation of logical equations is encountered when it is desirable for one reason or another to employ components and circuits which do not function in this straightforward step-by-step manner. Consider, for example, an ordinary binary counter consisting of four binary elements capable of counting to 16. As is well known, the counter can be made to function by causing the pulses to be counted to be applied to one of the four binary elements connected in a complementing circuit. When the binary value in this binary element changes from 1 to 0, a pulse is caused to be transmitted to a second binary element, also connected in a complementing circuit. Pulses derived in this manner are caused to be sent from one binary element to the next in a “ripple-carry” type of circuit. To assemble the counter, nothing more than

R. K. RICHARDS is a consulting engineer.

Richards—Logical Design Methods

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