Asynchronous Logics and Application to Information Processing

DAVID E. MULLER
University of Illinois

The limiting number of switching elements to which any given switching element may supply signals has been termed the "fan-out" of the element. Whenever logical design is undertaken, such fan-out restrictions must be observed by the designer, and the conditions imposed by these restrictions are usually considered quite separately from the logical conditions of the design. However, the speeds of logical elements in a switching system have long been known to bear an inverse relationship to their fan-out. This inverse relationship results from the fact that both speed and fan-out place conflicting requirements upon the power amplification of the element. In parallel computers, for example, one of the greatest limitations in speed occurs when signals from the control must be amplified and sent to all the gates in a register. This amplification is performed whenever a word is transferred from one register to another, and, in particular, it limits the speed of such operations as shifting and multiplication, which involve a large number of such transfers.

Circuit designers have recognized the fact that computation speed is often limited by the time required to amplify control signals and transmit them to the registers, and so have tended to concentrate upon the design of fast cable drivers and gate drivers rather than upon the design of flip-flops and other logical elements whose individual speeds have relatively little influence upon the speed of computation. On the other hand, logical designers and system designers have not usually been influenced by such considerations of the effect of fan-out upon speed, or at least not to the same extent, and the methods used for carrying out calculations and for performing arithmetic have remained unaffected.

In the present discussion a method of logical design will be proposed that is based upon the notion that fan-out is always limited and that calculation time depends upon the amount of amplification that must be performed. Although the method will be described from the standpoint of asynchronous logics, the basic problem is common to both asynchronous and synchronous systems, and a variation of the method may be derived that applies to synchronous systems. However, when the procedure is used for the design of an asynchronous system the methods used automatically ensure that the result will be speed-independent. Hence no special precautions need be taken during design to make certain that the course followed by the calculation does not depend upon the speeds of the logical switching elements.

Design, including logical design, is carried out by interconnecting standard logical blocks for transmitting and processing digits and control information. However, the standard logical blocks that are used are somewhat more complicated than the usual AND-, OR-, and NOT-elements from which logical design is usually begun. Instead the blocks are formed from several logical elements connected in a way characteristic of the block, and in general, in a way involving feedback so that the block has memory and is capable of retaining information.

Rather rigid rules will be set down for connecting logical blocks to form a larger circuit. This set of rules has been chosen in a somewhat arbitrary fashion but with the idea that it should be as simple as possible. It must, however, fulfill the dual purpose of ensuring both that the resulting circuit shall be speed-independent and that the fan-out shall be maintained below some specified value. Also, by careful choice of the set of interconnection rules one may reduce the design effort by simplifying the design procedure. Although the resulting circuit is of a rather special type, it fulfills the behavior specifications.

Each type of block in the set of standard block types is completely specified by

(1) listing and describing the logical decision elements (such as AND-, OR-, and NOT-elements) that make up the block, and listing the interconnections of these elements within the block; and

(2) designating a set of links by which the block may be connected to another block, and specifying the lines which comprise each such link.

A link, formed from a set of lines, makes possible the connection of the given block to another block. Each link of a block may contain several lines but will always contain at least one input line and at least one output line. An output line is taken as a line that is connected to the output of some element in the block; an input line is one that is not. Both input and output lines may be connected to the inputs of one or more elements in the block, although such connections do not always exist in output lines.

Whenever two blocks are connected together by means of some link, it is necessary for all input lines of the link of one block to be connected to output lines of the link of the other block and vice versa. Thus, if two blocks are to be connected together by a pair of links, the number of output lines of one link must equal the number of input lines of the other link. When such a connection is made the two links must be regarded as no longer available to other blocks for making other connections. Instead, any further connections to these blocks must be made through other links that are as yet unused.

The internal design of each block and the set of states through which it may pass are restricted in such a way that the signal changes (changes from 0 to 1 or from 1 to 0) that occur on the lines of any one link must occur in a well-defined sequence and must occur alternately upon input and output lines. Thus it is never possible for two changes to occur concurrently upon the lines of a given link, although it is possible for concurrent changes to occur upon two lines in different links. Also, after a change has occurred upon some line in a

given link the next change upon a line in that link must be upon a line passing in the opposite direction. However, these latter rules will be automatically fulfilled if the blocks are constructed and used properly, and the logical designer of the system need not be directly concerned with them.

After a circuit has been formed from blocks in this fashion, it will be capable of handling information by virtue of its local structure. Digits pass from block to block and remain uncoordinated except when they happen to impinge upon the same block. Thus digits carrying information will pass through the circuit in somewhat the same way as particles through a fluid, interfering only when they meet.

The rate at which information passes through a circuit of this type does not depend upon the size of the circuit, since all interactions are local. Thus one may increase the size of the circuit without sacrificing speed in digital operations. However, for the same reason it is not possible to use a conventional design with a centralized control of the usual type. Instead one must expect to determine the course of a calculation by directing the digits of the numbers involved to an appropriate portion of the machine. Compared to the conventional type of computer, a machine of this sort would tend to contain a greater number of specialized units for performing the various operations and would not contain the general-purpose devices (such as the adder-accumulator) that are common in machines today. Also, methods of performing arithmetic and other operations must be altered so that only a few digits interact at a time during the course of the operation.

As approached from this point of view, new meanings will be attached to the concepts of economy in time and equipment, since elementary logical operations can only be regarded as taking place between a few digits at a time, and executing large-scale parallel transfers of information under the continuous supervision of a centralized control becomes entirely pointless. However, parallel operations may be made to occur, but in a more spontaneous way after some initial direction from control signals. In a computer constructed in this fashion, one may expect to find logical operations occurring at the natural speeds of the logical elements rather than at the speeds of the drivers, which are fed by long lines from the control. Although a design of this type may realize an improvement in speed of as much as a factor of ten, the amount of equipment required is likely also to be much greater in terms of numbers of elementary decision elements.

Feasibility of systems of the type to be described here will be enhanced by the development, in the future, of more economical yet fast logic elements. Such elements, however, need not be connected to large numbers of control lines and therefore could be conveniently adapted to printed techniques. One difficulty would result from the use of conventional parallel high-speed memories with asynchronous block logic. In such memories, word transfers take place as a result of signals originating in the control, and hence these memories are poorly matched to the logic system to be discussed presently. A more natural way to handle the memory problem would be to construct the memory from logical blocks similar to those used in the other sections of the computer.

A large-scale memory of this type, however, would only be practicable if the logical blocks could be made very cheaply.

The choice of a particular set of block types will be determined by a variety of engineering considerations, but for purposes of illustration we shall select a particular basic set, which will be used throughout the remainder of this discussion. The use of this basic set should not be taken as an indication that the construction of loosely coordinated asynchronous circuits depends upon the specialized properties of a particular set of elements. The problem of choosing such a set of block types is similar in many respects to the problem of choosing a set of basic logical decision elements (AND, OR, NOT) to construct the blocks from. However, the number of block types required is greater than the number of decision element types, and the functions of the block types are more numerous. Furthermore it is probably not desirable from the standpoint of economy to use a minimum number of block types. Just as in the case of logical decision elements, an increase in the number of types above the minimum often allows one to construct simpler and more economical circuits.

Several types of logical decision elements will be used in the construction of blocks. Besides using AND- and OR-elements, we shall also make use of AND–NOT- and OR–NOT-elements, which are the same as the preceding elements but with inverted output. In addition to these we shall use *C*-elements, or memory devices that have two inputs and one output. The *C*-element retains its previous state as long as its two inputs do not agree with each other, but tends toward the state of the inputs whenever they both contain identical signals. We shall also imagine the availability of *C*-elements that have more than two inputs and are brought to the state of the inputs only when all lines agree.

The invention of a new type of decision element, the *C*-element, is not a necessity for the construction of the blocks. Although it is not possible to simply replace the *C*-element by its Boolean equivalent in terms of AND- and OR-elements and retain the same dynamic properties of speed independence for the resulting circuit, it is possible to dispense with the *C*-element. This is done by using more conventional flip-flops, constructed from the usual decision elements in circuits similar to, but slightly more complicated than, the ones described here.

Information within blocks is represented in the form of binary digits in a two-line system, using either 0 and 1 on the pair of lines or else 1 and 0, depending upon whether the binary digit being represented is a zero or a one. The pair 0 and 0 is used to separate digits, and is treated as a blank or spacer. A link between blocks that are used for passing information contains three lines; two of these lines go in the direction of information flow, and the remaining line used to indicate the acceptance of information goes in the opposite direction. A second type of link, which is used for passing signals that do not contain information, uses only two lines. As explained before, these lines must pass in opposite directions.

A typical block is shown in Fig. 1. The two links at the left-hand side of the block have three and two lines each and the link on the right-hand side has three

lines. However, the three-line link on the left has two inputs and one output, whereas the link on the right has two outputs and one input. Evidently information passes from left to right through this block. The three-line link on the right may be connected to another link of the type appearing at the upper left but not to another link having two outputs and one input.

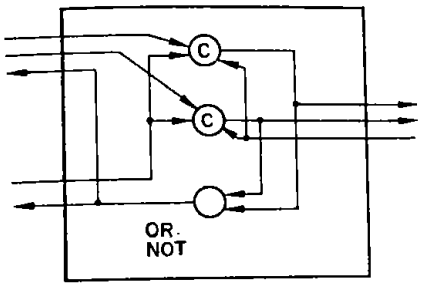

FIG. 1

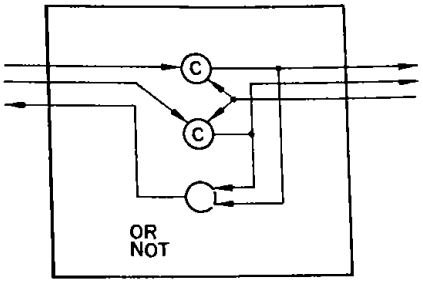

FIG. 2

In the quiescent state, both *C*-elements of this block are in the 0 condition. Information is brought into the block by the application of a 1 signal to one of the input lines on the upper left link, and a 1 signal at the input of the lower left link. The information passes through the upper link, and the lower link supplies a regulating signal. If a 1 signal also is present at the input of the right-hand link, then one of the *C*-elements is enabled to pass into the 1 state. This signal at the right-hand link indicates that the block to the right of this link has received the spacer that was originally present in the block under discussion. After the information has passed into the present block, a completion signal is formed at the output of the OR–NOT-element and passes to the left as a 0 signal. The presence of this signal permits the next spacer to enter the two blocks connected to the left-hand links.

A block type that is similar but does not contain the link on the lower left is shown in Fig. 2. By connecting blocks of this type together, we may form a chain of blocks in which two lines pass from left to right and one from right to left in each connecting link. If the initial state of the system is chosen in such a way that all OR–NOT-elements are in equilibrium and all binary digits represented on pairs of lines are separated by spacers, then the result is speed-independent. The total number of binary digits does not change during the action of the circuit, and the order and identity of the individual digits are also retained, but the digits tend to pass to the right through the chain of blocks until, in the final equilibrium condition, the digits all lie in the rightmost blocks, separated by single blocks containing spacers. Thus, the digits in the line of blocks "fall" to the right through the line of blocks in the way that balls would roll through an inclined pipe, passing as far and as rapidly to the right as they can without interfering with the digits ahead of them.

The complete set of blocks using *C*-elements is shown in Fig. 3. An abbreviated symbolism is shown at the right of each block in which single lines are used to represent links. Within each block the *C*-elements are numbered, and within each link the *C*-elements to which the lines of the link are connected are

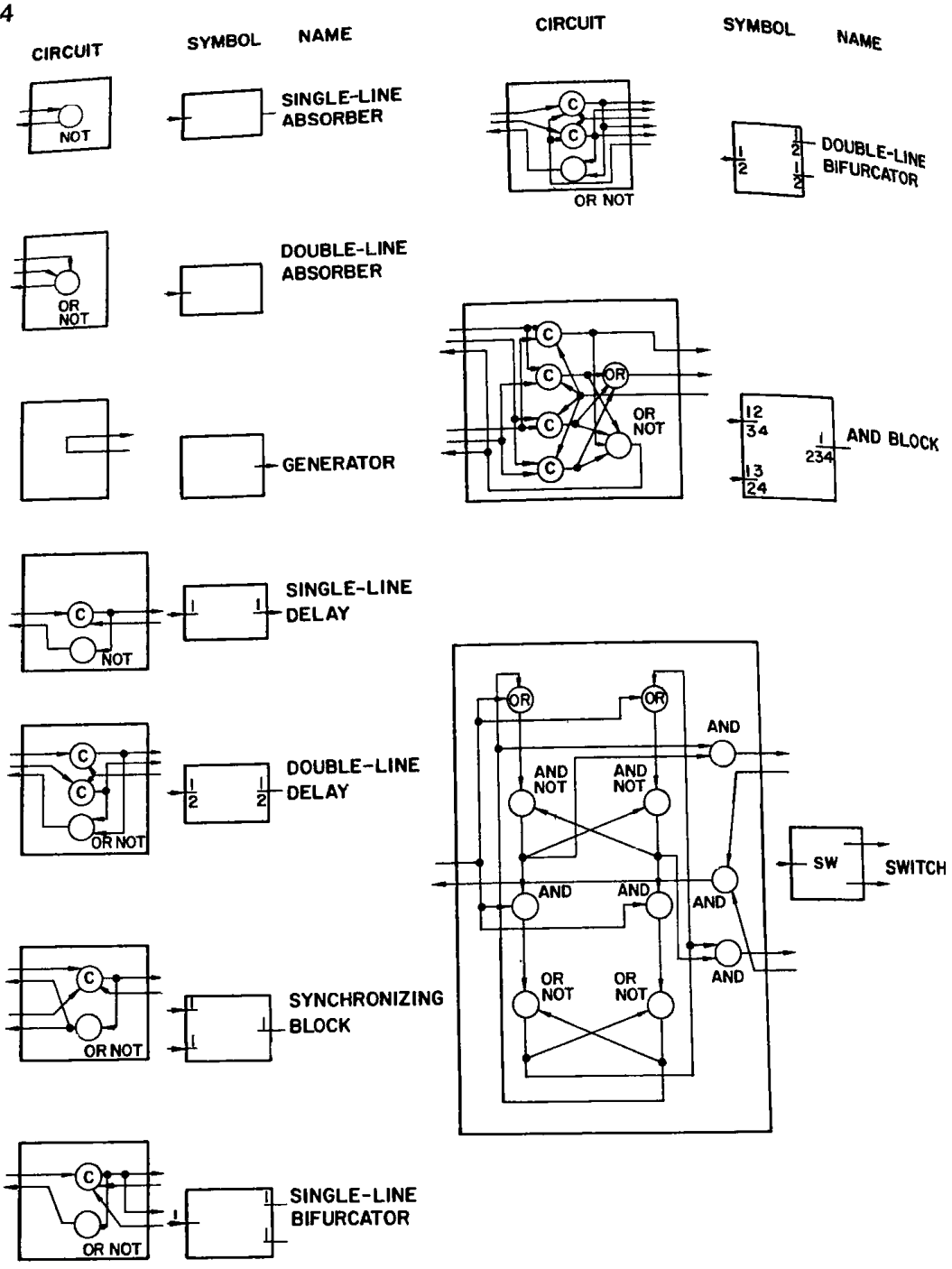

FIG. 3

indicated in the abbreviated symbolism by numbers written next to the link. If the link has two lines passing in the same direction, then the numbers of the *C*-elements connected to the top line are written above the numbers of the *C*-elements connected to the bottom line. It is always assumed that all the *C*-elements connected to two lines passing in one direction are connected to a single line passing in the opposite direction. Input lines to the block are connected directly to the *C*-elements, whereas output lines are connected through an OR-element if they pass to the right and through an OR–NOT-element if

they pass to the left. An arrow is placed on each link in a direction counter to the direction of the NOT-element associated with the link. This is also the direction of information transfer when the link contains three lines.

The set of blocks has the property that any pair of links on any pair of blocks may be connected together, provided that the directions of the arrows on the two links agree (left hand links may only be connected to right hand links and vice versa), and provided that the two links contain the same number of lines. If the connection rules are followed and if the initial state of the circuit is chosen in accord with the binary digit and digit spacer convention described earlier, the result will be a speed-independent circuit.

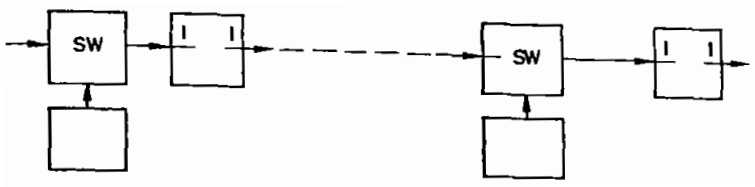

FIG. 4

To carry out the design of a useful circuit consisting of these blocks it is necessary to know how they function as conveyors and processors of information. Logical operations between digits are provided in the AND and exclusive-OR blocks and in combinations of these using appropriate interchanges of lines within links to carry out inversions when necessary. The correct channeling of digits is designed into the circuit with the help of the various types of switches and the bifurcating block, which duplicates the digit that passes into it.

Only one digit may be present in any one block at a time and, in view of the fact that pairs of digits must always be separated by spacers, we see that no more than about half the blocks may hold distinct digits at one time. Even fewer digits than this are actually present because if the digits are to move through the circuit we must provide blocks for duplication of digits and spacers during transition. In fact, if one assumes that all blocks undergo changes in approximately the same length of time, then the maximum information processing rate is obtained if there are two blocks for each digit and two blocks for each spacer. In this case half of the blocks are in equilibrium while half are undergoing transition at any given time. Owing to the fact that the number of digits present in the circuit varies as a function of time one cannot expect to obtain this optimum information processing rate in practice, and usually more than half of the blocks are in equilibrium.

Three examples of basic circuits are given in which the blocks in use are those taken from the basic set shown in Fig. 3. The first example (Fig. 4) is a counter that uses three types of blocks. Although the counter itself does not process binary digits, it may be used to control the flow of digits to various other blocks. After receiving a signal at its input it delivers 2^n similar signals at its output. Such action is accomplished by using switch blocks that feed alternate inputs to the output. Thus queues of signals waiting at the two inputs are merged and alternately fed to the output.

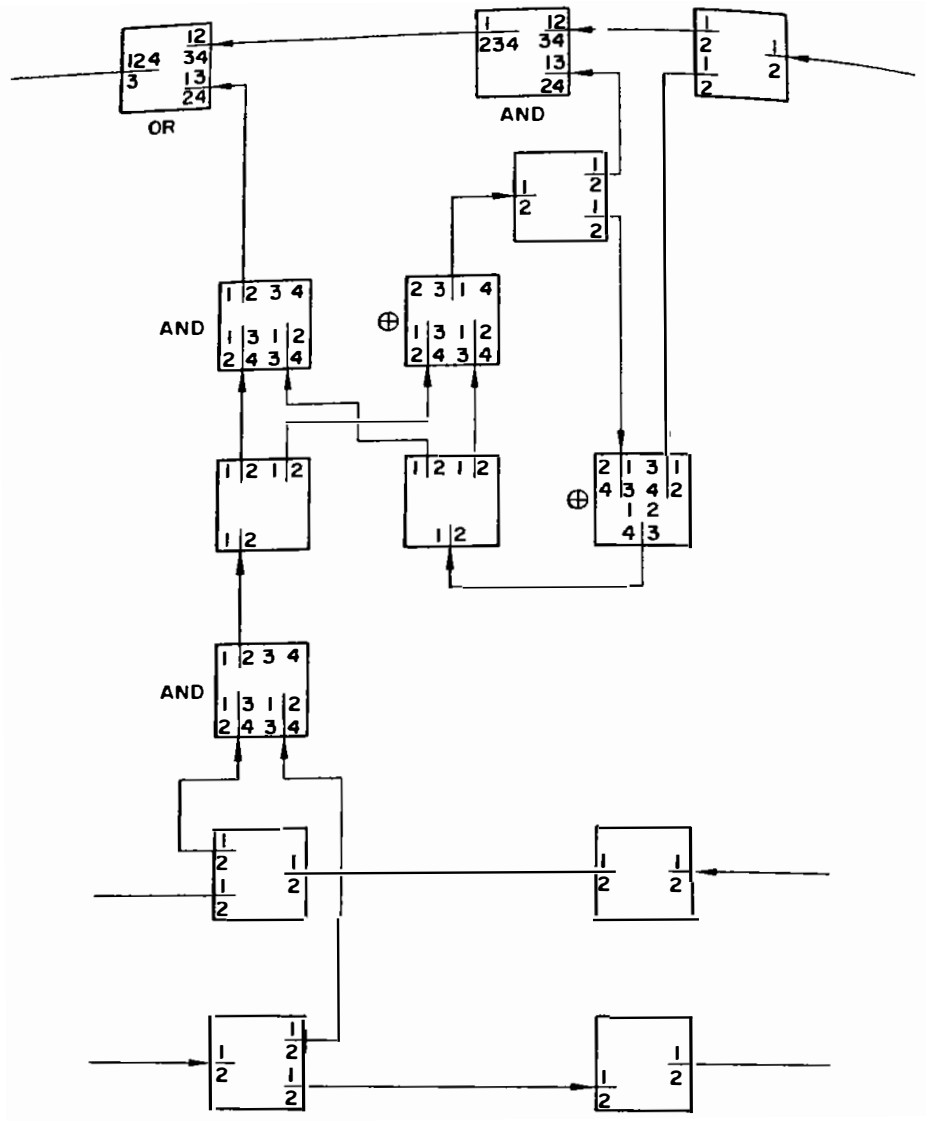

FIG. 5 One stage of a multiplication circuit.

Figure 5 is a sketch of a multiplier unit composed of blocks that carry digits. It is impossible to assign a completely definite order to the processing of the digits of the numbers being handled. Yet such order does exist with respect to the events that occur at any one position in the circuit.

The multiplier and the multiplicand pass through the two chains of delay and bifurcating blocks at the bottom of the figure. The multiplicand passes to the right as the multiplier passes to the left. Since it is impractical in the present system for the individual multiplier digits to operate upon the entire multiplicand at once, we chose the present arrangement, in which digit-by-digit combinations of multiplier and multiplicand occur. A product is accumulated in the blocks lying above the two bottom chains. As each digit of the product is accumulated, a carry digit (which in this circuit may be either a 0 or a 1) is generated and fed into the next higher digit position. Carries in this circuit pass to the left at the same rate that multiplier digits pass to the left. Parallel combination of the digits of the multiplier and multiplicand occurs in such a way that the difference

in significance of any pair of combining digits is the same at any instant, assuming all blocks have about equal speed. After both multiplier and multiplicand have passed from their respective chains the completed product will have been formed.

It is interesting to note that the delay blocks placed between stages at the bottom of the multiplication circuit are necessary for the correct operation of the circuit. Without these delay blocks the circuit would fail completely to operate, and "traffic jam" conditions would exist among the digits, causing the digits to stop flowing, except in a slow, serial fashion. Clearly, if systematic algebraic (or other) rules are formulated for designing such circuits, it will be necessary to take such problems into account.