"The objectives of the mixed logic system are to achieve simplicity, clarity and applicability of schematic diagrams used to represent logic designs."

Mixed Logic Systems

by
Rene Dos Remedios*

ABSTRACT

This paper presents, in an introductory manner, a method of creating and documenting digital circuit schematic diagrams. The method emphasizes a technique of generating schematic diagrams which maximizes their semantic content. This is achieved through the use of "natural" logic operators and through a convention which relieves the designer from having to think about final implementation details while designing the circuit. This paper also aims to show that mixed logic is a viable alternative to the present methods of teaching digital electronics. Unfortunately, the method, though widely accepted in industry is not given its due prominence by the academe.

I. INTRODUCTION

When we talk of digital electronics, it is difficult to conceive of the development of such a large body of human knowledge without the use of the mathematics of logic algebra and its application to switching theory. In logic algebra, simple thought processes are represented mathematically. The systematic application of logic algebra to switching circuits allows designers to tackle problems that would have otherwise been too unwieldy to handle. Of course, the great strides in the development of digital electronics would not have been possible without the discovery of semiconductor devices and the miniaturization of electronic circuits.

The first treatise on logic algebra, "An Investigation of the Laws of Thought on which are Founded the Mathematical Theories of Logic and Probabilities," was written in 1854 by George Boole, an English mathematician. This work became the foundation of logic algebra which eventually became better known as Boolean Algebra. It is interesting to note that Boole did not relate any of his work to electrical or mechanical devices.

In 1938, at the Massachusetts Institute of Technology, Claude Shannon, who was later to become better known for his contributions to information theory, wrote a paper on how an electrical circuit composed of switches and relays, could be represented mathematically by Boolean algebra. This work stood as a milestone in the development of switching theory and its application to digital electronic circuits.

*Faculty member, Department of Electrical Engineering, U.P. College of Engineering, Diliman, Quezon City, Philippines.

Switching theory, or rather, its application, is mainly responsible for the practicality of many present day applications that would have been too complicated to implement without a systematic mathematics as an analysis tool. Switching theory, however, simply deals with the truthfulness or the falsity of a proposition, whether, some variable is a 1 or a 0. This aspect of the theory leads to the obvious choice of assigning a discrete voltage level in a circuit to the value of a logic 1 and assigning another voltage level to a logic 0. Thus, an electronic circuit can be made to model certain thought processes with great accuracy and repeatability. For instance, in a positive logic system, a logic 1 might be represented by +5 volts and a logic 0 represented by 0 volts. On the other hand, a negative logic system would assign 0 volts to logic 1 and 5 volts to logic 0. Based on one of the previous conventions a circuit designer can then build a circuit designed to implement some logic function, where the response of the logic function to input stimuli can be observed by simply measuring voltages at some point in the circuit.

In a real system, the assignment of logic values to voltage levels is usually not so absolute. Assignment of logic values to circuit voltage levels usually constitutes a range of values. For example, in the positive logic CMOS system, logic 1 could be assigned values between 3v and 5v and logic 0 could be assigned values between 0v and 2v. As another example, for positive logic TTL systems, a logic 1 corresponds to voltages between 1.8v and 3.5v. Logic 0 corresponds to voltages between 0v and 0.8v.

This state of events would have been adequate if the assignment of true or false to a given voltage level had not been so arbitrary. One designer would arbitrarily assign a high voltage level to activate a digital device while another designer could, just as arbitrarily, assign a low voltage level to activate the same device. In other words, some designers would choose positive logic and others negative logic in implementing logic functions. The problem was compounded by the abundance of digital devices using medium scale to large scale integrated circuit technology (MSI and LSI) with a multiplicity of inputs and outputs, each being assigned a positive logic or negative logic level with unpredictability.

Actually, many of the seemingly arbitrary assignments of voltage levels are based on sound engineering principles. For example, it is well known that TTL logic is more easily affected by noise when inputs are at a low voltage level. Designers usually assign the normally unselected state to the higher voltage level to minimize the probability of errors due to false triggering because of noise. Because of this, most MSI or LSI devices have their control inputs activated when the voltage level is at or near 0 volts (a negative logic system). This means that most of the time the inputs remain at a safe, noise resistant 5 volts. When a signal has no normal logic state, that is, half of the time the signal is high and the other half of the time the signal is low, by convention, designers normally assign the positive logic system to that signal. We can see, therefore, why some devices have both positive and negative logic signal levels.

In spite of the confusion of logic levels, ordinary switching theory still serves its role as an analysis and synthesis tool, but many designs are becoming increasingly difficult to implement because of the indiscriminate use of both positive and negative logic in the same digital system. Such designs often result in documentation nightmares because of the absence of a standard convention. A designer is often unable to explain the functionality of a signal at a certain point in the circuit because of the uncertainty on whether he is dealing with a positive or negative logic signal.

One solution to this dilemma would be the adoption of a standard positive (or negative) logic system. This, unfortunately, would be near to impossible because of previously mentioned conflicts in engineering judgements. A system has been adopted and advocated by various electrical and electronics organizations such as the IEEE and the IEC. The system is commonly known as Mixed Logic.

MIXED LOGIC

The objectives of the mixed logic system are to achieve simplicity, clarity and applicability of schematic diagrams used to represent logic designs The benefits would be reduced design effort, increased understandability of schematic circuit designs and an overall reduction in system development costs.

Mixed logic emphasizes only a few concepts yet the advantages a logic designer would gain from its use far outweigh the effort involved in learning a new design documentation system.

The following are the main concepts emphasized by the mixed logic system:

1. The concept of assertion levels versus positive or negative logic;
2. The use of logic functions versus the use of logic gates; and
3. The use of mnemonics in naming logic variables.

The use of assertion levels separates the implementation method from the design method. Simply stated, a variable is asserted when the logic condition associated with the particular variable is true. For example, if you have a digital line X, which determines when lamp A is turned on, then, you can say that when the control line or variable X is asserted the lamp A turns on. On the other hand, you may have a line Y, which determines when lamp B is to be turned off. Thus, when Y is asserted, lamp B will turn off.

Assertion of a variable is associated with the logical function of the variable and not the voltage level found at some point in a circuit. Because of this simple concept, the design as prescribed by switching theory or some similar problem description method becomes independent of the hardware implementation. This is not true in positive or negative logic systems.

If a device to be controlled is activated when its control input is at high voltage level, then a variable or control line which is also asserted when high will be required to control the line. If a variable is asserted at a high voltage level and it turns out that there is a control input that requires a low voltage for assertion, all that is required is a voltage level changer or a voltage inverter. Notice that the voltage inverter does not change the logic function of the control line, it only changes the assertion level of the line.

Many conventions exist for the indication of assertion levels. One method which is advocated by William Fletcher (1980) is the use of H or L enclosed in a parenthesis immediately after the logic variable to indicate the high and low assertion levels, respectively.

In Figure 1, LAMPON (H) and LAMPON (L) refer to the same boolean variable LAMPON. When control inputs C (H) and D (L) are asserted the corresponding lamps turn on. The inverter in the circuit does nothing to change the logic function of the variable LAMPON. It is better to think of the inverter as a matching device which matches the assertion level of LAMPON (H) with that of control input B (L). The introduction of "matching inverters" is reserved for the final implementation and in no way comes into consideration during earlier design phases.

Since understandability of the schematic diagram is of prime importance, it is also very important to document circuitry using only logic functions of common language usage. These functions are the AND, OR, and NOT logic functions. Sometimes the exclusive or (EXOR) or the equivalence function is used.

It is not natural to think in terms of NAND and NOR logic functions and so these functions are not used in the development of designs and the accompanying schematics. For example, one can say it is natural to think, "I will watch a movie if Jose AND Juan will also watch the movie."

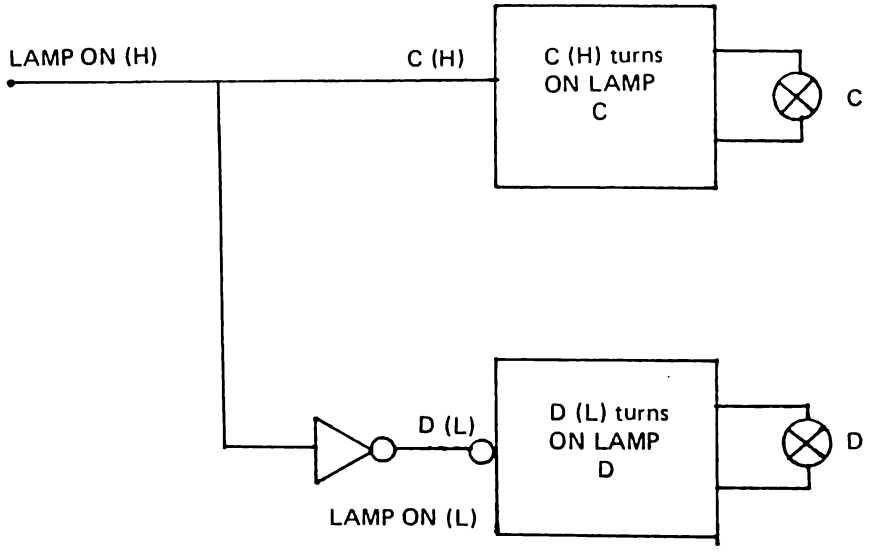


Figure 1

Another reasonable possibility is to say, "I will watch the movie if Maria OR Rowena will watch the movie with me." Few people will understand you, however, if you say, "I will watch the movie if Jose NAND Maria will watch the movie." Of course, you may grammatically rephrase the statement to make it meaningful, ". . . if Jose AND Maria will NOT watch the movie." Notice the operators used in rephrasing are NOT and AND which, taken together, is of course the same as NAND. The semantic content of NOT and AND is clear but that of NAND is not.

We must also distinguish a logic function from a digital IC logic gate. A logic function serves the role of expressing a designer's idea of the circuit functionality. A logic gate is a digital device which may be used to implement a logic function in hardware. The shapes used in schematic diagrams to implement various logic functions at different assertion levels are given in Figure 2 together with possible hardware implementations using different logic gates. Notice that a small circle is used to indicate that inputs or outputs are asserted low.

A standard convention of representing logic functions and assertion levels has been proposed in IEEE standard 91-1984. This standard, however, has not achieved popular support. Most designers still use the symbols given in Figure 2 in spite of the close similarities to positive logic symbol conventions. Some of the relevant symbols in IEEE standard 91-1984 are presented in Figure 3.

In Figure 4 is an example of a mixed logic and a positive logic system used to document the same digital system. The lines and the gates used are exactly the same. The logic function implemented can be stated as: an ACCumulate OR a BLaNK OR a CLocK together with DATA, causes a Load DATA in some register.

If we look at the mixed logic diagram and interpret it in terms of the depicted logic function, it is clear that the mixed logic circuit more clearly describes the functionality of the given circuit.

The above example also illustrates the third important aspect of mixed logic. It is apparent that the use of mnemonics in naming the lines on a schematic diagram can do a lot towards making the design clearer and easier to understand. One of the objectives of mixed logic is to obtain self documenting schematics. With a judicious use of mnemonic identifiers, together with the clarity afforded by the use of only the abovementioned logic functions, self-documenting mixed logic circuits are not uncommon.

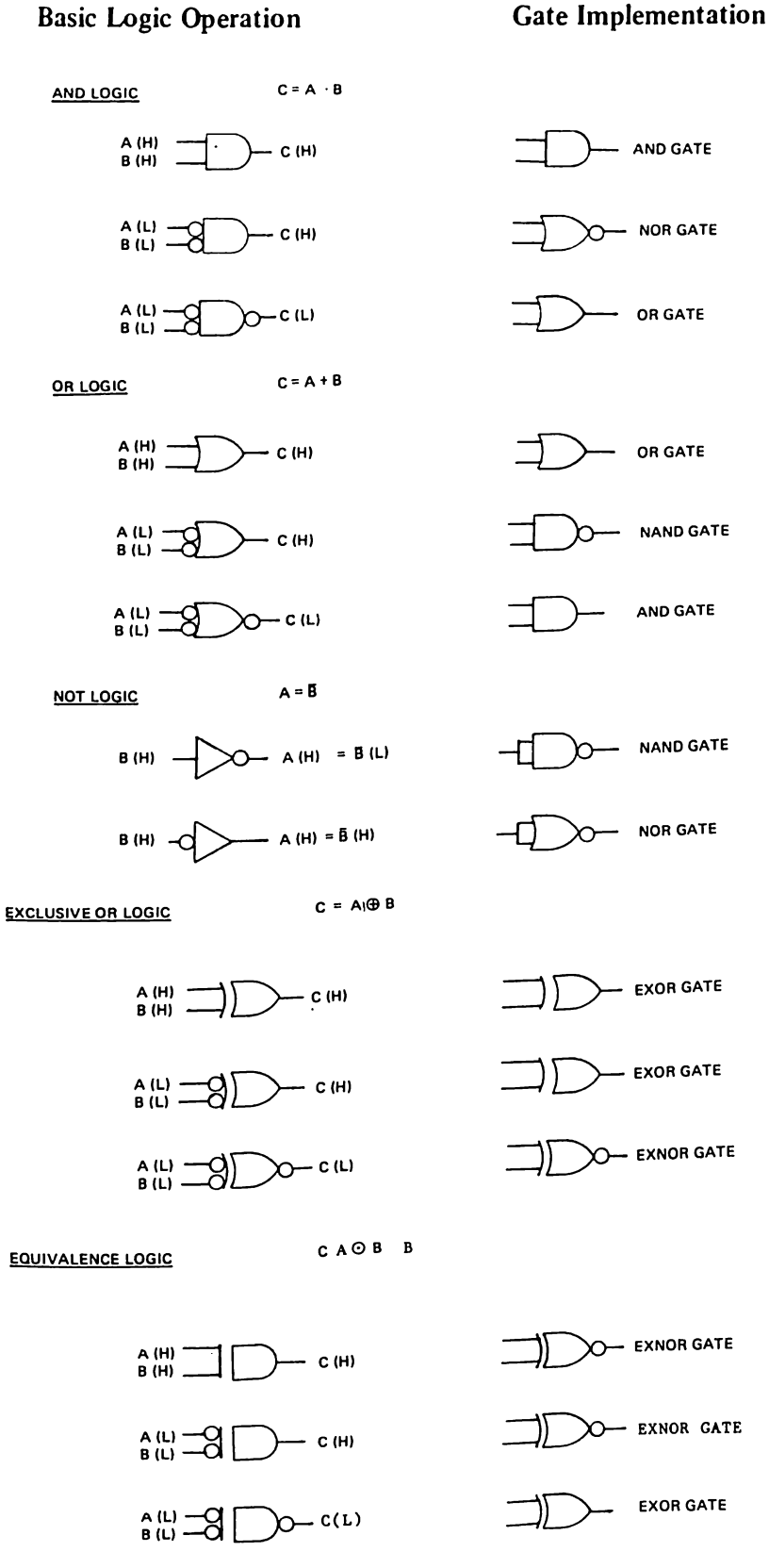

Basic Logic Operation
Gate Implementation
AND LOGIC
C = A · B
A (H)
B (H)
C (H)
AND GATE
A (L)
B (L)
C (H)
NOR GATE
A (L)
B (L)
C (L)
OR GATE
OR LOGIC
C = A + B
A (H)
B (H)
C (H)
OR GATE
A (L)
B (L)
C (H)
NAND GATE
A (L)
B (L)
C (L)
AND GATE
NOT LOGIC
A = B̄
B (H)
A (H) = B̄ (L)
NAND GATE
B (H)
A (H) = B̄ (H)
NOR GATE
EXCLUSIVE OR LOGIC
C = A⊕ B
A (H)
B (H)
C (H)
EXOR GATE
A (L)
B (L)
C (H)
EXOR GATE
A (L)
B (L)
C (L)
EXNOR GATE
EQUIVALENCE LOGIC
C A⊙B B
A (H)
B (H)
C (H)
EXNOR GATE
A (L)
B (L)
C (H)
EXNOR GATE
A (L)
B (L)
C(L)
EXOR GATE

Figure 2

SOME IEEE STANDARD 91 – 1984 Symbols

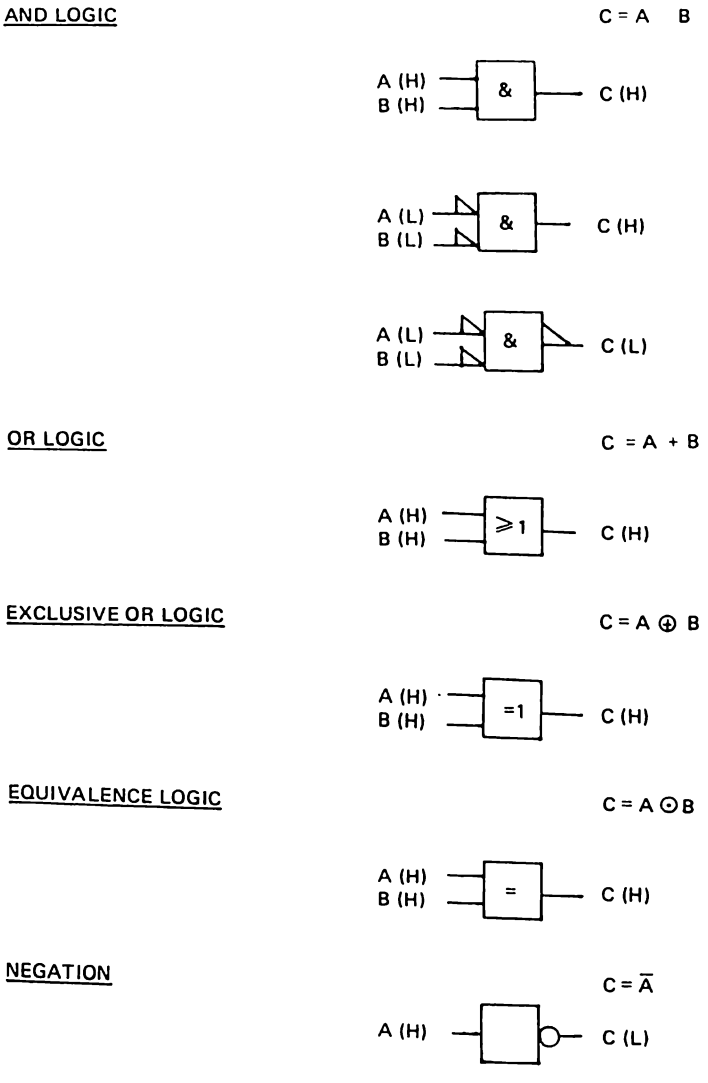
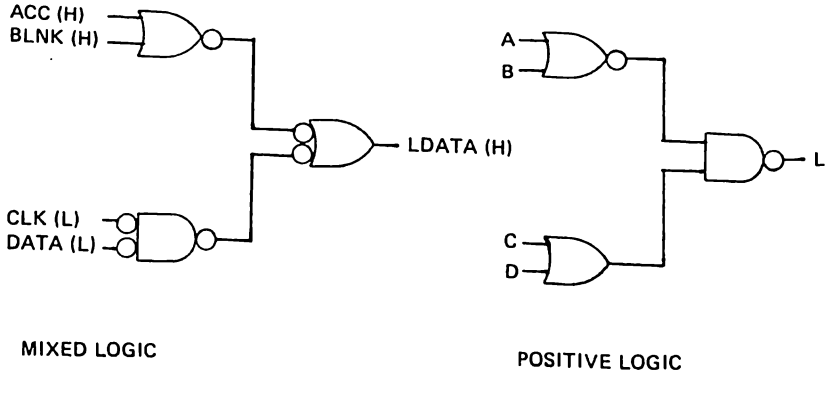


Figure 3



LDATA = (ACC + BLK) + CLK DATA

Figure 4

Lastly, another reason why mixed logic circuits are becoming more popular is because circuits documented using mixed logic are far easier to implement using real devices than circuits not documented with mixed logic. This is because the mixed logic circuit is not hardware implementation dependent. It becomes a simple matter to implement a circuit using NOR gate or NAND gate designs if the circuit is documented by mixed logic. (See Figure 5.) As we can see in the example, the implementation of the basic logic functions does not necessarily have to be implemented using the same gate type. For example, the OR logic function in the figure can be implemented using either NAND gates or NOR gates. Of course, the OR logic function can also be implemented using an OR gate.

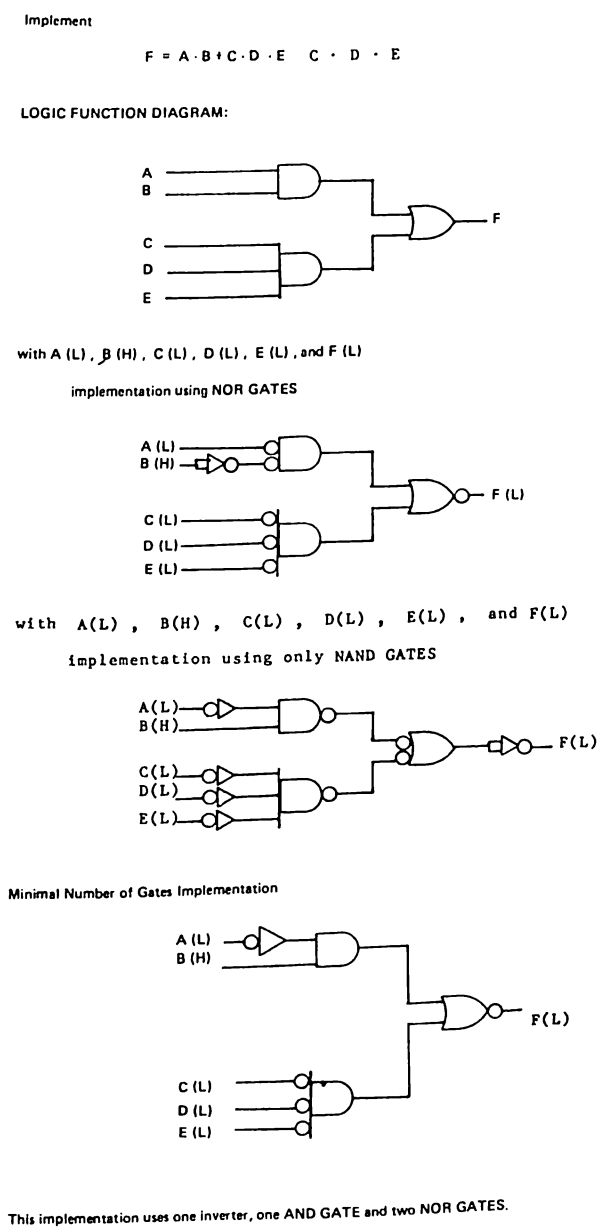


Figure 5

Mixed logic can be used to great advantage by following a few simple rules:

1) Always draw the logic functions in the schematic diagrams, not the logic gates.

2) Watch out for conflicts in assertion levels. A line must always begin and end at the same assertion level to preserve the value of the variable. If a line does not meet the above requirement, an inverter gate may be used to change voltage levels without changing logic levels.

3) If a line does not begin and end at the same assertion level, a logic inversion has taken place. If an inversion in logic is then desired in a matched line, an inverter may be introduced to cause the level mismatch.

4) As much as possible, try to make all inputs of a gate agree in assertion level. To this end, two inversion circles may be added to a line without changing the logic levels in the circuit. However, the gates used in the implementation will change, in general.

Figure 5 is an example of the implementation of the function F=AB+(CDE) under different conditions of variable assertion. The function is implemented using NOR gates only, then, using NAND gates only, and finally, using a mixture of gates to obtain a minimal number of gates result.

Notice that in this example all the lines begin and terminate at the same assertion level. This is because there is no logic inversion anywhere in the original function. We also see in this example the use of mixed logic to obtain the minimal number of gates implementation. Minimization of the number of gates required to implement a logic function is achieved by the straightforward method of absorbing as many inverters as possible into a gate structure. This procedure illustrates rule 4, above, concerning the objective of trying to make all inputs of a gate agree in assertion level. In general, if there is disagreement in assertion levels among the inputs of a gate, extra inverters will be required.

CONCLUSION

Mixed logic is actually used by many large corporations for documentation and design purposes. Examples would be IBM, INTEL, National Semiconductor Devices, and Motorola. Unfortunately, the prominence given the system by industry seems to be unnoticed by most educators. Only a few textbooks in digital design have been written using mixed logic exclusively in the text. Notable examples of the use of mixed logic are Fletcher (1980) and Prosser and Winkel (1981). It is reassuring to see though, that more of the newer texts have at least mentioned the system and its merits. It is hoped that in the future, more importance will be given a good design and documentation method-mixed logic.

REFERENCES

FLETCHER, W. I., *An Engineering Approach to Digital Design.* Englewood Cliffs, N. J.: Prentice Hall, Inc., 1980.

PEATMAN, J. B., *Digital Hardware Design.* New York: Mcgraw Hill Book Co., 1980.

WAKERLY, J. F., *Logic Design Projects Using Standard Integrated Circuits.* New York: John Wiley and Sons, Inc., 1976.

WINKEL, D. and F. PROSSER, *The Art of Digital Design.* Englewood Cliffs, N. J.: Prentice Hall, Inc., 1981.

IEC Technical Committee, Publication 117-15, 1970.

IEEE Standard 91-1984.