

A Case for Teaching Mixed Logic in Digital Design

David L. Livingston

Department of Electrical Engineering, Virginia Military Institute

I. Introduction

An important aspect of any logic design problem is an easy and direct design method which results in properly documented components; i.e., schematics, expressions, truth tables, etc., which accurately convey the designer's original thought process. The mixed logic approach, which meets the above requirements, is proposed as a method to be taught in digital design courses.

To understand mixed logic, the concepts of positive and negative logic and assertion levels must first be defined. Positive logic is defined as a high voltage level representing a logic 1 and a low voltage level representing a logic 0. Negative logic is the reverse, i.e., a low voltage level represents a logic 1 and a high voltage level represents a logic 0. Assertion levels determine whether a net or signal line in a digital circuit is to be interpreted as positive logic or negative logic. A negative logic assertion level is called active low and is represented by the presence of bubbles on the net in the schematic and a .L suffix on the logic variable. A positive logic assertion level is called active high and is represented by the absence of bubbles on the net in the schematic and a .H suffix on the logic variable. Mixed logic is the use of both positive and negative logic representations in a digital circuit design.

Many authors of introductory digital texts mention mixed logic – sometimes directly, often times as a graphical technique – but use the positive logic approach almost exclusively^{1,2}. It is the purpose of this paper to show that a mixed logic approach is superior to a strictly positive logic approach and discuss the pedagogy of mixed logic in digital design courses. A very brief discussion of mixed logic principles is presented in the sequel. A complete coverage of mixed logic can be found in Prosser and Winkel³ and Breeding⁴.

The remainder of the paper is organized as follows. Section II emphasizes the differences between logic functions and logic gates. A synthesis procedure is presented and compared to a positive logic approach with an example in section III. In section IV the analysis of mixed logic circuits is discussed. The pedagogy of mixed logic is discussed in section V and conclusions are drawn in section VI.

II. Functions Vs. Gates

In a mixed logic approach to digital design, it's important to make a distinction between logic *functions* and *gates*. Logic functions compose the underlying mathematical representation

or model that forms the basis of the solution to a given design problem, whereas gates are the physical implementation of the solution. Logic functions are generally represented in three ways: 1) as a set of algebraic expressions over logical variables, 2) in symbolic form using the standard symbols for AND and OR and slashes for NOT, and 3) in the form of a truth table using 0 and 1 for false and true, respectively. Note, the only functions required are AND, OR and NOT – XOR and MATCH functions can also be used where appropriate. NAND and NOR *functions* are not necessary in a mixed logic approach.

Gates used in mixed logic design include AND, OR, NAND, and NOR gates and inverters. The names for the gates are derived from the positive logic interpretation; it can be shown that a negative logic interpretation results in the dual functions. To understand the mixed logic approach, it is important to learn the transfer characteristic or voltage table for each gate. The voltage tables for the five basic gates are shown in Table 1 – L is used to represent low voltage and H high voltage and the functional symbols are used for the positive logic interpretation. Note that an inverter is not necessarily a NOT gate; it simply changes assertion levels.

A	B	AB	A+B	A B	A↑B	A'
L	L	L	L	H	H	H
L	H	L	H	H	L	H
H	L	L	H	H	L	L
H	H	H	H	L	L	L

Table 1. Voltage tables for five gates.

The logic function that is implemented by a particular gate depends on the assertion levels of the input and output signals. For example, if all i/o signals are active high, an AND gate implements the AND function as shown in Table 2. However, if all i/o signals are active low, the same AND gate implements the OR function as shown in Table 3. Note that the voltage tables are the same for both Table 2 and Table 3.

NAND and NOR gates should be used to implement only AND and OR functions and inverters. These are the cases from which mixed logic gets the “mixed” part of its name. To use a NAND gate to implement an AND function, the assertion levels of the input signals are active high and the assertion level of the output signal is active low. To use a NAND gate to implement the OR function, the assertion levels from the AND function should be reversed. The two cases are demonstrated in Tables 4 and 5. Note that again, the voltage tables for both functions are the same being that the NAND gate is used in both cases. NOR gates are used in a dual fashion.

A.H	B.H	C.H
0, L	0, L	0, L
0, L	1, H	0, L
1, H	0, L	0, L
1, H	1, H	1, H

Table 2. AND function using AND gate.

A.L	B.L	C.L
0, H	0, H	0, H
0, H	1, L	1, L
1, L	0, H	1, L
1, L	1, L	1, L

Table 3. OR function using AND gate.

A.H	B.H	C.L
0, L	0, L	0, H
0, L	1, H	0, H
1, H	0, L	0, H
1, H	1, H	1, L

Table 4. AND function using a NAND gate.

A.L	B.L	C.H
0, H	0, H	0, L
0, H	1, L	1, H
1, L	0, H	1, H
1, L	1, L	1, H

Table 5. OR function using a NAND gate.

Gate symbols should represent both logic and assertion levels. For example, an AND gate used to implement an AND function would use the standard AND symbol as shown in Figure 1. If the same gate is used to implement the OR function, the symbol used should be the OR symbol with bubbles to indicate the negative logic assertion levels as shown in Figure 2.



Figure 1. AND function using AND gate.



Figure 2. OR function using AND gate.

III. Synthesis

The synthesis of digital systems using multiple types of gates including NAND and NOR gates is an area in which the mixed logic approach surpasses the positive logic approach. Using positive logic only for designs with NANDs and NORs requires the repeated use of double complements and de Morgan's Laws. This results in a significant amount of algebraic manipulation which is error-prone. The steps for designing a digital circuit using mixed logic follow. The process will be illustrated with an example that compares both methods.

Mixed Logic Synthesis Method

- 1) Create a logic diagram directly from logic expressions using AND and OR symbols and a slash for NOT.
- 2) Determine the assertion levels of all input and output signals adding bubbles to each active low signal.
- 3) Convert AND and OR symbols to gate symbols by adding bubbles and add a bubble to one side of each slash, attempting to create matching bubble pairs where possible.
- 4) Insert inverters for all unmatched bubbles.
- 5) Optionally remove slashes.

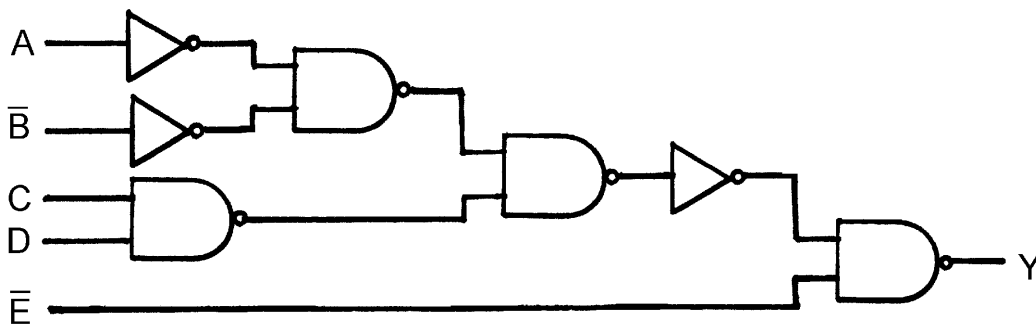
Example

Design a circuit for the following expression and assertion levels using 2-input NAND gates and inverters only.

$$Y = (A'B + CD) + E; Y.H, A.H, B.L, C.H, D.H, E.L$$

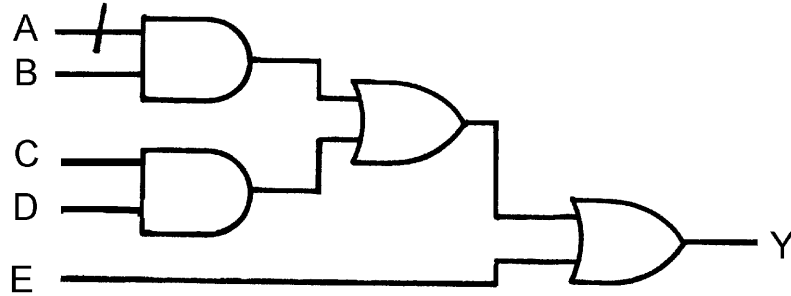
Positive Logic Solution

- 1) For positive logic we must complement the B and E variables since they are active low: $Y = (A'B' + CD) + E'$
- 2) Manipulate into NAND functions:
 $Y = (A'B' + CD) + E' = (A'B' + CD)' + E'$
 $Y = ((A'B')'(CD)')' + E' = ((A'|B')(C|D))' + E'$
 $Y = ((A'|B')|(C|D)) + E'$
 $Y = (((A'|B')|(C|D))' E')' = (((A'|B')|(C|D))'|E)$
- 3) Draw schematic using NANDs and inverters.

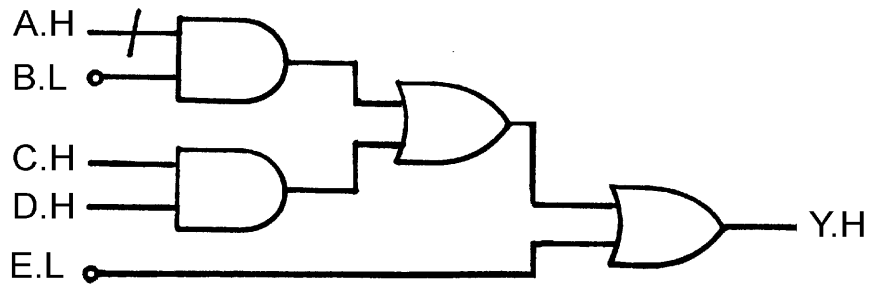


Mixed Logic Solution

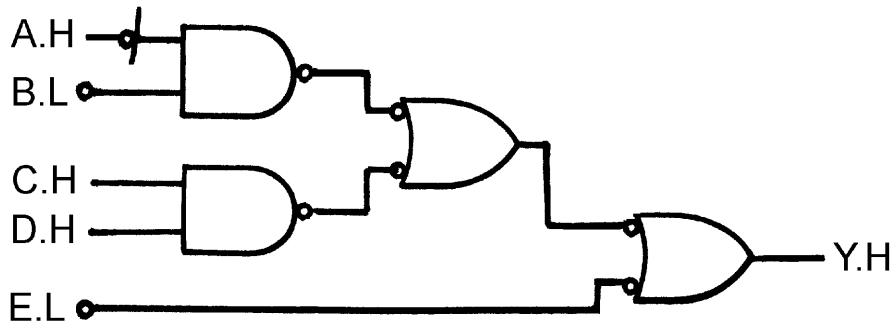
- 1) Draw the symbolic representation using AND and OR symbols and slashes for NOT.



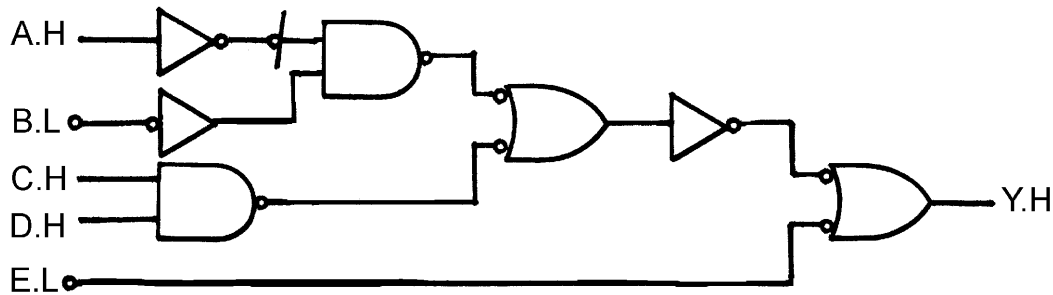
- 2) Add bubbles to all active low signals.



- 3) Add bubbles to the outputs of AND symbols and inputs of OR symbols to change them to NAND gates. Add a bubble to one side of each slash attempting to create bubble pairs.



4) Insert inverters where necessary to create bubble pairs.



Comparing the results of both the positive logic implementation and the mixed logic implementation, the resulting schematics represent the same physical circuit. However, the original logic expression is clearly represented by the mixed logic circuit.

IV) Analysis

As with synthesis, the use of mixed logic has an advantage over positive logic when analyzing a digital circuit. This is particularly true when obtaining the logical expressions from the schematic diagrams. Using positive logic requires the repeated use of de Morgan's Laws for circuits with NAND and NOR gates and inverters.

Analyzing a mixed logic circuit is straight-forward. If the schematic has had slashes removed, insert them at every place where there is an unmatched bubble. At this point, simply ignore the bubbles and write the logical expression directly from the diagram interpreting slashes as the NOT function. Note, inverters are simply assertion level translators from positive-to-negative or negative-to-positive logic and can be ignored.

The logic circuit can be easily evaluated for constant inputs using both logic values and voltage levels. If a logic value is known, the voltage level is easily determined from the assertion level. For example, a logic 0 at a net where there is a pair of bubbles results in a high voltage level, since the assertion level is active low.

V) Teaching Mixed Logic

The introduction of the mixed logic method should occur at the time implementations are discussed. This generally happens after covering standard forms of logic expressions and possibly after minimization with Karnaugh maps. It's certainly appropriate to start mixed logic when NAND and NOR gates and multilevel implementations are presented.

An appropriate presentation schedule is as follows:

- 1) Motivation of developing a logic design method which is straight-forward and self-documenting.
- 2) Definitions of assertion levels, positive, negative and mixed logic.
- 3) Survey of various gates, inverters and buffers, their associated voltage tables, and use for the three primitive logic functions.
- 4) Synthesis.
- 5) Analysis.
- 6) Conversion from positive logic to mixed logic.

Of utmost importance is stressing of the difference between logic functions and logic gates. This is somewhat complicated by the use of positive logic function names for gates. Wakerly explains duality by demonstrating that both AND gates and OR gates can be used to implement both AND and OR functions². He uses the terms "type 1" and "type 2" gates to try to dispel the confusion between functions and gates. The difference between functions and gates can be reinforced in graphical presentations by pointing to a symbol and asking students what gate the symbol represents and what logic function is being implemented.

Several problems can occur with teaching the mixed logic approach. If a student has prior experience with positive logic, mixed logic can be confusing. A particular point of confusion is the fact that inverters are not always NOT gates. This can be cleared by drilling the concept of assertion levels as they're related to logic levels and voltage levels. Another point of confusion is between assertion levels and voltage levels. This occurs due to the use of H and L for high and low voltage levels and as suffixes for positive logic and negative logic, and the use of the terms active high and active low. This might be rectified by using indicators other than high and low, such as positive and negative, to label assertion levels.

VI. Conclusions

The mixed logic method produces results which are easily obtained. Since mixed logic is a graphical approach, students are less likely to make errors in the design process as they would if a large amount of algebraic manipulation is involved. The results of the method also reflect the original thought process of the designer. Therefore communication between members of a design team is facilitated. These benefits warrant the teaching of mixed logic in digital logic design courses.

References

1. M. Mano, *Digital Design, 2nd Ed.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1991.
2. J. F. Wakerly, *Digital Design, Principles and Practices, 3rd Ed. Updated.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 2001.
3. F. P. Prosser and D. E. Winkel, *The Art of Digital Design, An Introduction to Top-Down Design, 2nd Ed.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.
4. K. J. Breeding, *Digital Design Fundamentals, 2nd Ed.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1992.

DAVID L. LIVINGSTON

David Livingston is an Associate Professor of Electrical Engineering at Virginia Military Institute. He has also been employed as a computer engineer for IBM Endicott Labs and was on the faculty at Old Dominion University and Virginia Western Community College. Dr. Livingston received the B.S.E., M.E. and Ph.D degrees in Electrical Engineering at Old Dominion University. He is a licensed professional engineer in the Commonwealth of Virginia and is a member of IEEE, ASEE, HKN, and the Skeptics Society.