Name: Anders Pitman

Course: CSC120

Section: #33198

Instructor: Bruce Carlton

Lab Number: Software Lab #1

Date Due: 2/19/2015

Date Submitted: 2/19/2015

Objectives:

1. For the student to become familiar with LogicWorks software.
2. For the student to learn about designing and composing digital circuits, including a half adder, 1 bit full adder, 4 bit full adder, and 4 bit incrementer.

Expected Outcome:

1. The student will successfully implement and test all specified circuits in LogicWorks.
2. The student will gain a deeper understanding of the circuits involved.

Task #1 - Building a Half Adder:

A half adder takes 2 separate 1-bit inputs and outputs a sum and carry (COUT). The sum represents the result of adding the 2 bits. However, since there is only 1 bit sum output, both inputs being set to 1 will result in the carry output being set to 1. The truth table is shown in Table 1.

Input		Output	
A	**B**	**COUT**	**SUM**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 1 - Half Adder Truth Table

The SOP boolean equations are:

SUM = A'B + AB'

COUT = AB

The circuit schematic for a half adder is shown in Figure 1.

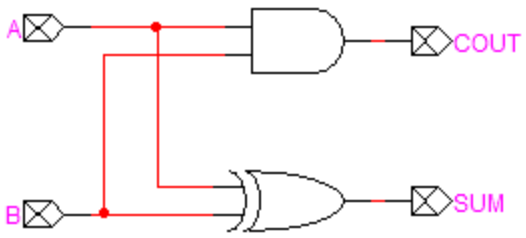

Figure 1 - Half Adder Implementation

A gate level test case for row 4 of Table 1 is shown in Figure 2

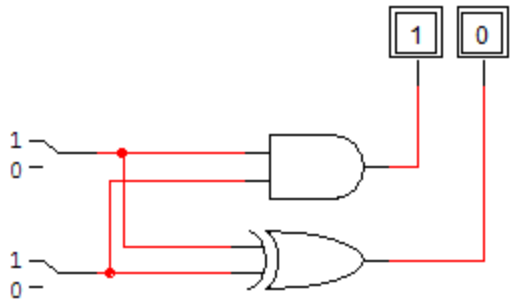

Figure 2 - Test Case Half Adder Row 4

When creating circuits using design software such as LogicWorks, it is very useful to create discrete components with abstract implementation details, which can be copied and reused. LogicWorks calls these “device symbols”. The device symbol for the half adder is shown in Figure 3. This device symbol operates exactly as the gate level implementation above, but is represented in a more concise and abstracted manner.

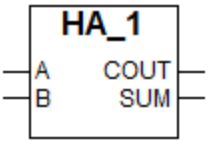

Figure 3 - Half Adder Device Symbol

Another example test case for a half adder, taken from row 2 of Table 1, covers the case where input A=0 and input B=1, and the output should be COUT=0 and SUM=1. This test case is shown in Figure 4.

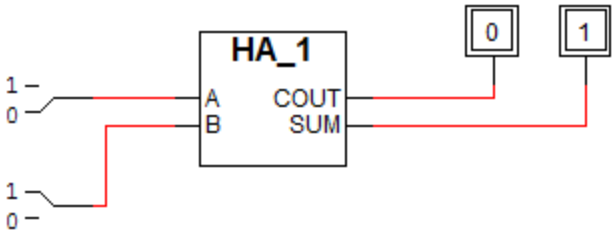

Figure 4 - Test Case Half Adder Row 2

Task #2 - Building a 1-Bit Full Adder:

The 1-bit full adder adds the result of 2 input bits, similar to a half adder. In fact, the full adder can be implemented as a combination of half adders. However, the full adder offers an improvement over the half adder in that it also handles a carry input (CIN), which is used to indicate if there is a 1 being carried from any previous components in the circuit. Exploitation of this trait will be shown later in the implementation of the 4 bit full adder. The truth table for a 1 bit full adder is shown in Table 2.

Input			**Output**	
A	**B**	**CIN**	**COUT**	**SUM**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 2 - 1-Bit Full Adder Truth Table

The boolean equations are as follows:

SUM = (A XOR B) XOR CIN

COUT = AB + (A XOR B)CIN

The circuit schematic for the gate level implementation of the 1 bit full adder is shown in Figure 5.

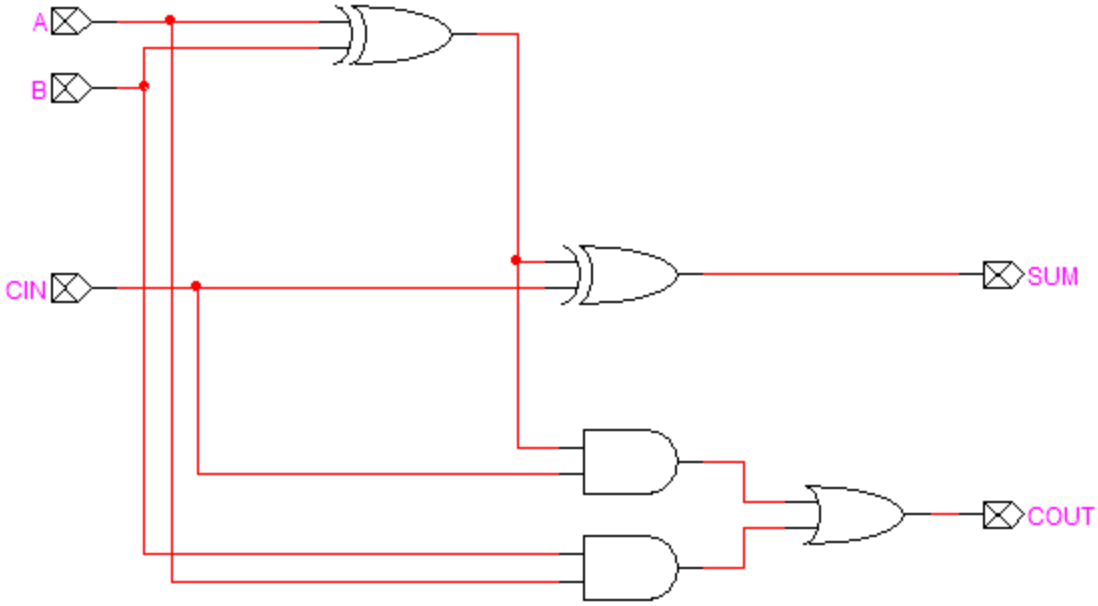

Figure 5 - Gate Level Implementation of 1-Bit Full Adder

A gate level test case from row 5 of Table 2 is shown in Figure 6.

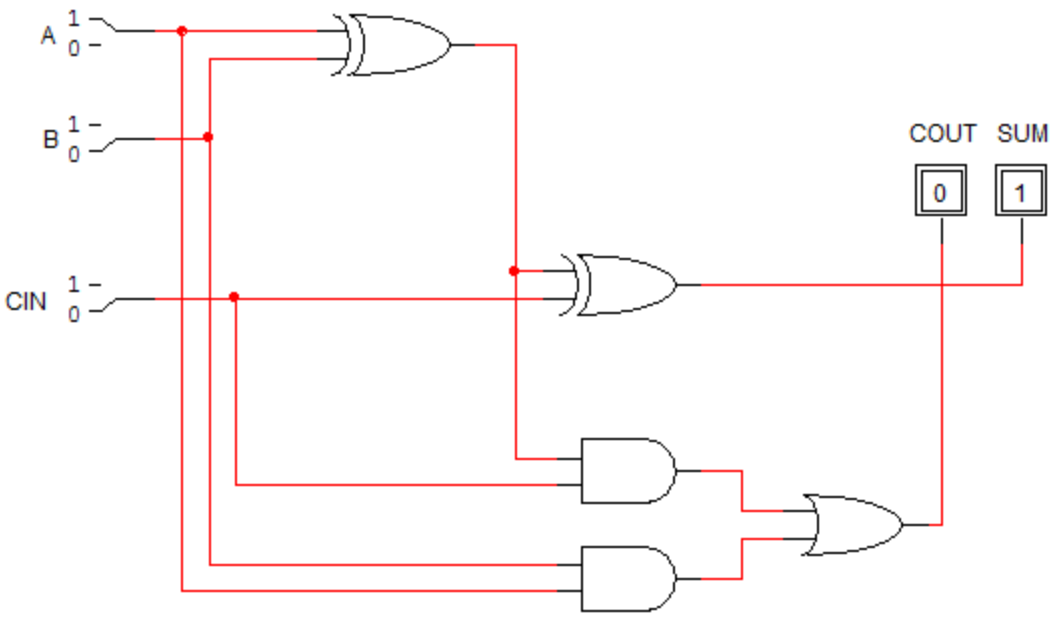

Figure 6 - Test Case 1-Bit Full Adder Row 5

As mentioned previously, the 1-Bit full adder can also be implemented using half adders. The circuit schematic for the implementation of the 1 bit full adder using half adders is shown in Figure 7.

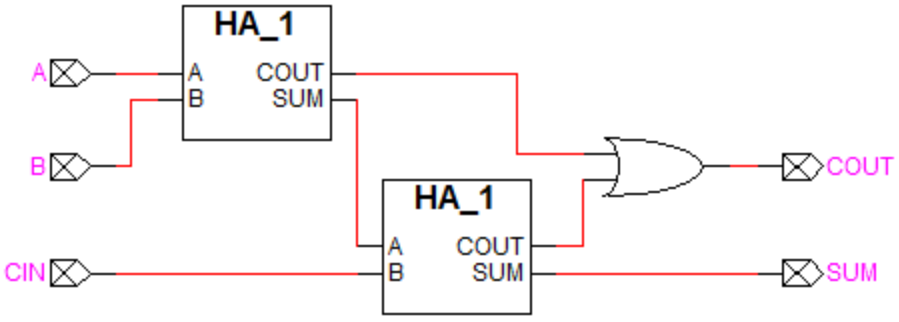

Figure 7 - Full Adder Implementation From Half Adders

The device symbol for the 1-Bit full adder is shown in Figure 8.

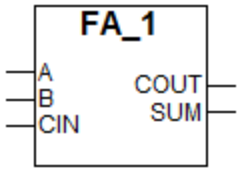

Figure 8 - 1-Bit Full Adder Device Symbol

A symbol level test case of the 1-bit full adder, taken from row 7 of the truth table, is shown in Figure 9.

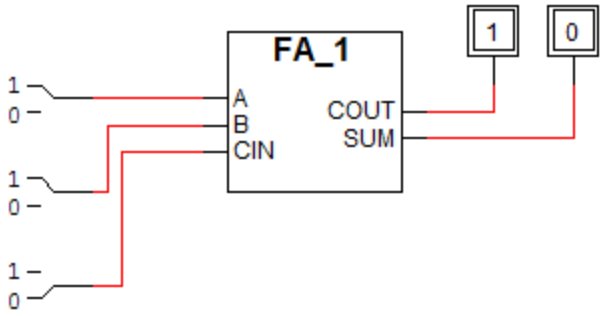

Figure 9 - Test Case 1-Bit Full Adder Row 7

Task #3 - Building a 4-Bit Full Adder

A 4-bit full adder adds 2 4-bit binary numbers together. If the output exceeds 4 bits, the COUT carry will be set to 1. The 4-bit adder is implemented by chaining 4 1-bit full adders together. The boolean equations are as follows:

S0 = (A0 XOR B0) XOR CIN

C1 = A0*B0 + (A0 XOR B0)CIN

S1 = (A1 XOR B1) XOR C1

C2 = A1*B1 + (A1 XOR B1)C1

S2 = (A2 XOR B2) XOR C2

C3 = A2*B2 + (A2 XOR B2)C2

S3 = (A3 XOR B3) XOR C3

COUT = A3*B3 + (A3 XOR B3)C3

This is more clearly expressed in the circuit schematic in Figure 10.

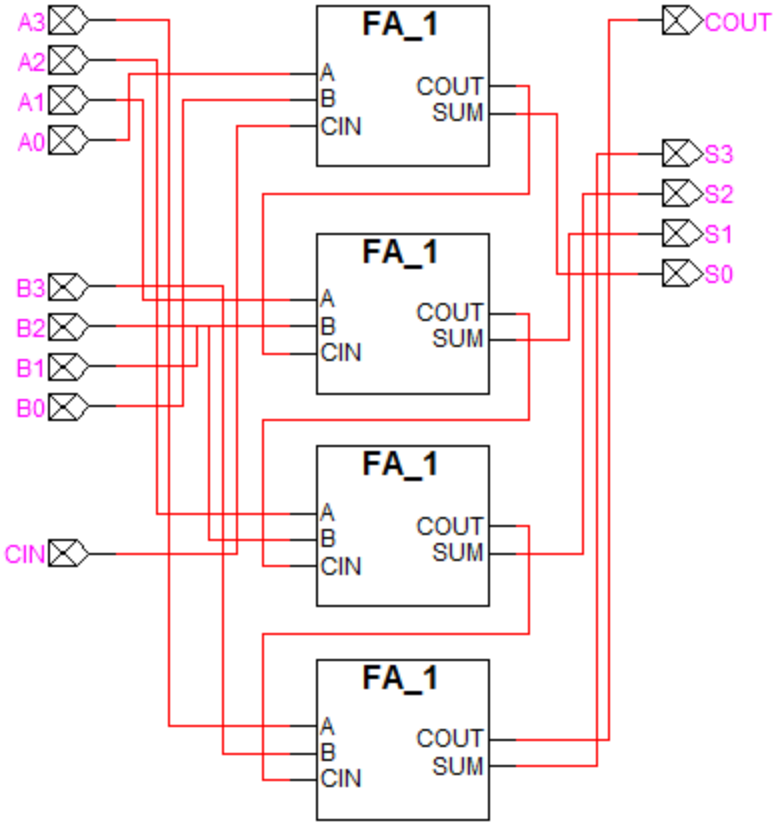

Figure 10 - 4-Bit Full Adder Implementation

The test case in Figure 11 shows the addition of 3 and 9. The output is the hexadecimal number C (decimal 12), as expected. Since 12 can be contained by 4 bits, COUT is 0.

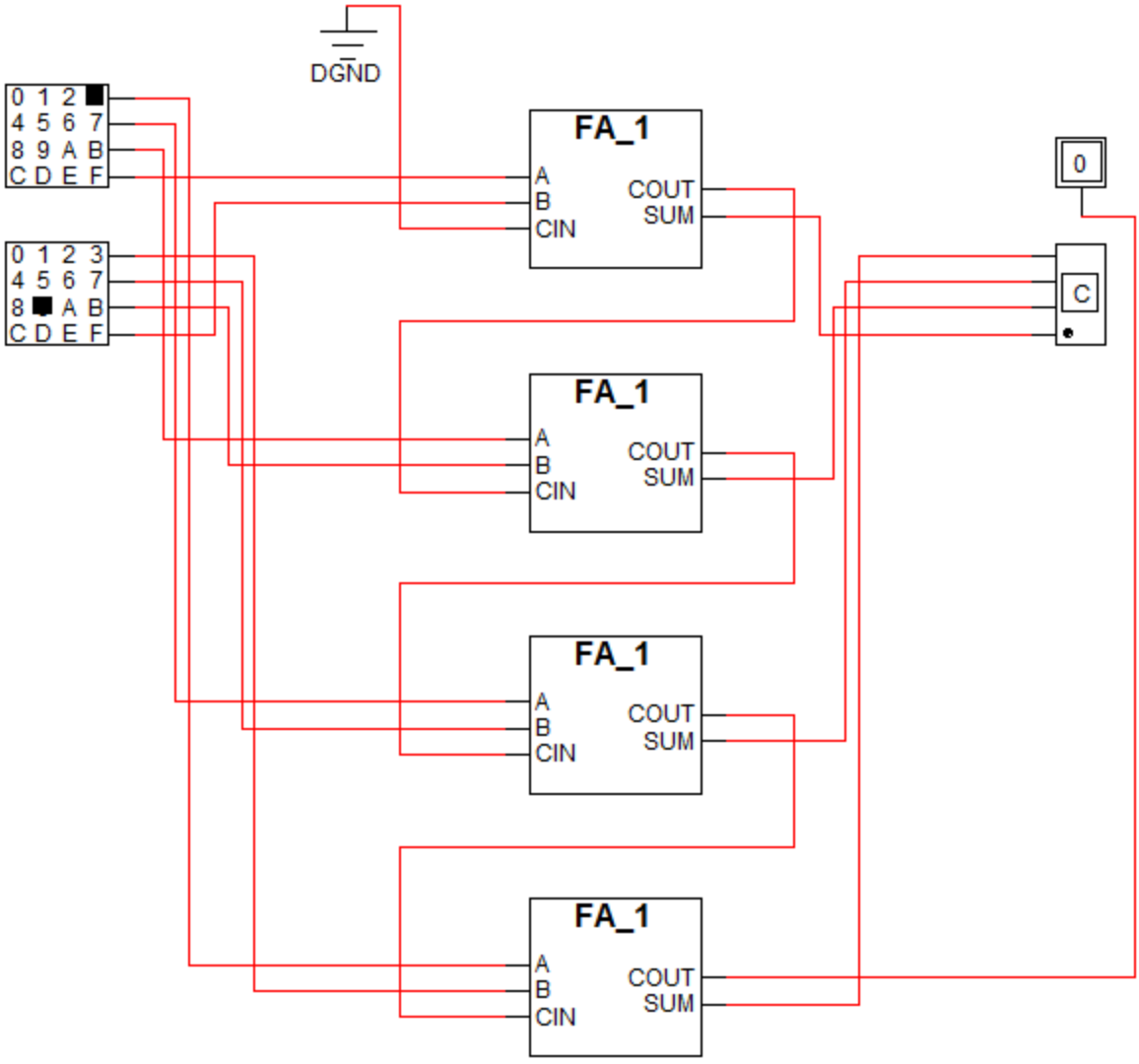

Figure 11 - Test Case Add 3 and 9

The device symbol of the 4-bit full adder is shown in Figure 12.

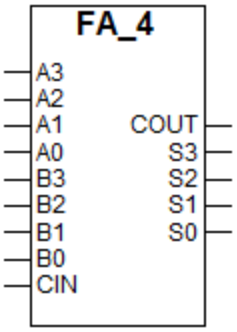

Figure 12 - 4-Bit Full Adder Device Symbol

The test case of adding 7 plus 9 is shown in Figure 13. In this case, the resulting value is hexadecimal 10 (decimal 16), as expected.

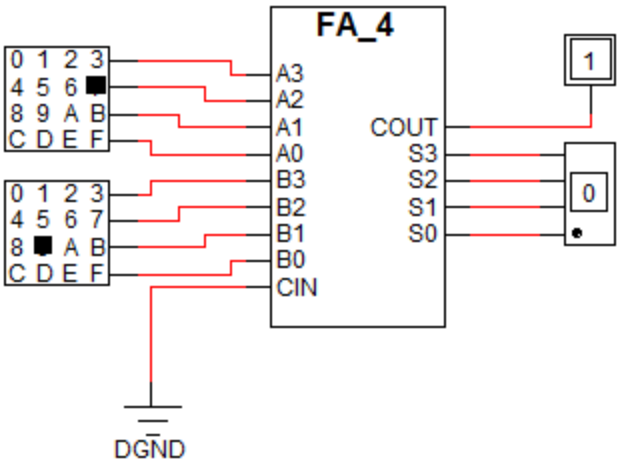

Figure 13 - Test Case Add 7 and 9

Task #4 - Building a 4-Bit Increment

The 4 bit increment takes a 4 bit number as input, increments the value by 1, and outputs the incremented value, carrying via COUT if the output exceeds 4 bits. The 4 bit increment is implemented using 4 half adders, as shown in Figure 14.

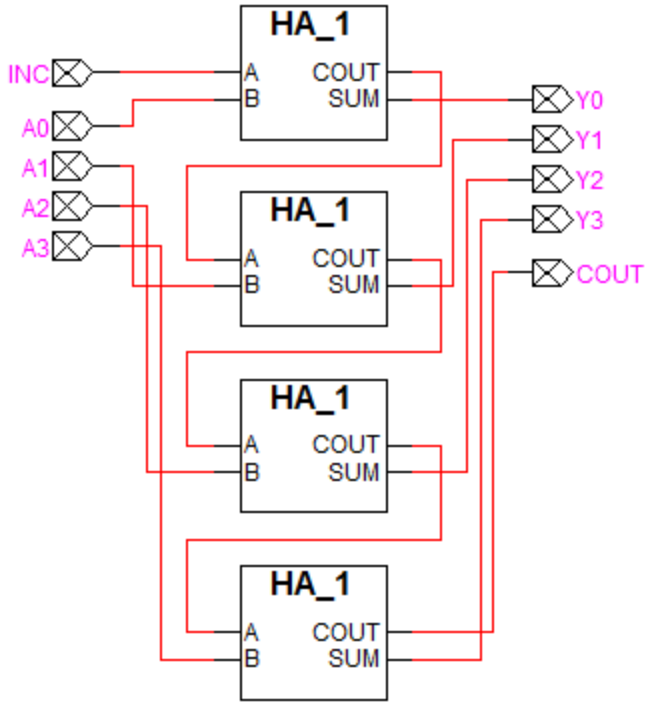

Figure 14 - 4-Bit Increment Implementation

A test case showing the incrementation of the number 5 is shown in Figure 15.

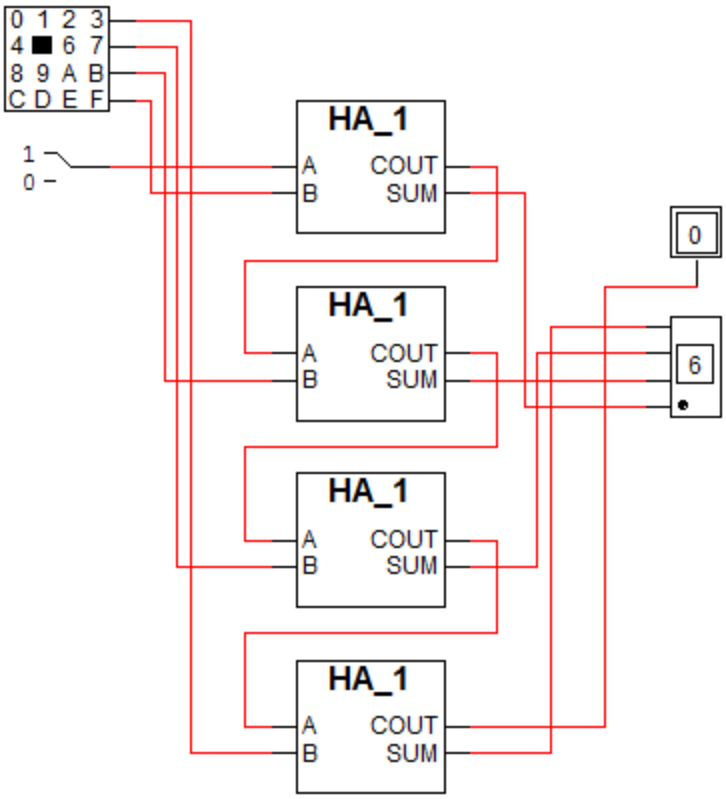

Figure 15 - Test Case Increment 5

The device symbol for the 4-bit increment is shown in Figure 16.

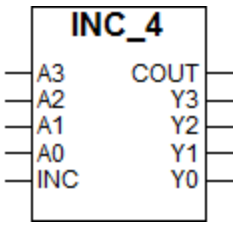

Figure 16 - 4-Bit Increment Device Symbol

A test case incrementing the value 9 using the device symbol increment is shown in Figure 17.

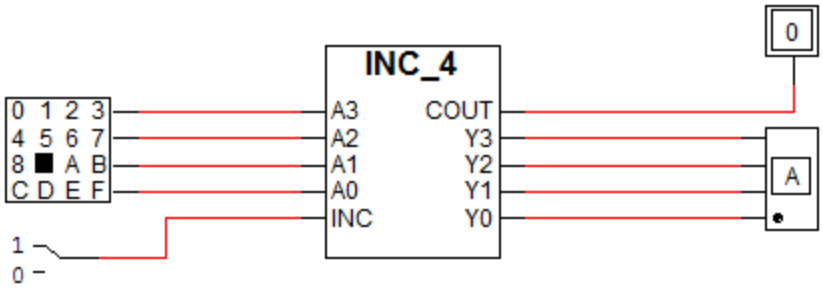

Figure 17 - Test Case Increment 9

Observations:

1. The student gained much valuable experience with LogicWorks, including creating circuits and saving as .cct files, creating a new library and saving as a .clf file, creating new device symbols and saving within a library, and using existing and student-created devices from libraries.
2. The student successfully created all of the specified circuits, and an increased level of understanding was attained.

Technical Comments

1. The student did not encounter any serious technical problems. The LogicWorks software was intuitive and reliable.