CMPE 160 Laboratory Exercise 6

Binary Addition and Subtraction Circuits

Andrei Tumbar
Performed: February 20th
Submitted: February 27th

Lab Section: 4
Instructor: Mr. Byers
TA: Sam Myers
Kobe Balin
Georgi Thomas

Lecture Section: 1
Professor: Mr. Cliver

Your Signature: ________________________

Abstract

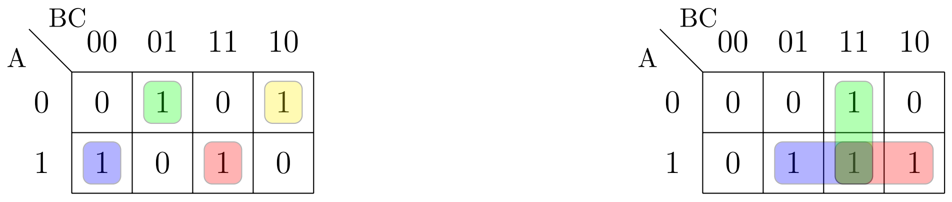
In this laboratory exercise a binary adder/subtractor was implemented using the 2's complement implementation of subtraction. First a full adder was implemented by creating a Karnaugh Map and then reducing the boolean expression. A full adder's output consists of a sum and a carry which will go into the next full adder. K-Maps and boolean expressions were written for both. The circuits were created in Quartus and simulated in ModelSim. A VHDL test bench was provided for input signal control. Following the creation of a single full adder, a four-bit full adder was then given. By using a control signal and two four-bit binary numbers, an adder/subtractor was created in Quartus. Both the full adder and the 4-bit adder/subtractor was implemented on the breadboard. This exercise was successful in creating a circuit that adds and subtracts arbitrary inputs.

Design Methodology

To first implement a full adder, a Karnaugh Map must be formed. Then, will the resultant boolean expression, a circuit diagram can be formed.

Full Adder

Two Karnaugh Maps were created to implement both outputs of the adder. The letters A and B denote the two binary numbers being added while C denote the carry in of the full adder.



(a) K-Map of sum function.

(b) K-Map of carry function.

Figure 1: K-Maps for both outputs of the full adder.

None of the groups in Figure 1a overlapped meaning that no boolean reduction from the k-map can be completed. In Figure 1b three different groups are created. A min-term expression can be extracted from these k-maps. These terms can then be reduced to use fewer gates.
The letter F will be used to express the full adder function. Therefore to distinguish each output F_S will denote the sum and F_C will denote the carry. A, B, and C will be the inputs from the Karnaugh maps.

$$F_S = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C} \tag{1}$$

$$F_C = AC + BC + AB = A(B + C) + BC \tag{2}$$

All of the min-terms in the Figure 1a were written as a boolean expression. Next, using the rules of boolean, F_S (equation 1) can be simplified. Equation 2 shows the min-terms of F_C and a simple distributive simplification performed to reduce the total number of gates used.

$$\begin{aligned}
F_S &= \bar{B}(A\bar{C} + \bar{A}C) + ABC + \bar{A}B\bar{C} && \text{Distributive property}\\
&= \bar{B}(A \oplus C) + ABC + \bar{A}B\bar{C} && \text{Definition of XOR}\\
&= \bar{B}(A \oplus C) + B(AC + \bar{A}\bar{C}) && \text{Distributive property}\\
&= \bar{B}(A \oplus C) + B(\overline{\overline{AC + \bar{A}\bar{C}}}) && \text{Double negation}\\
&= \bar{B}(A \oplus C) + B(\overline{\overline{AC} \cdot \overline{\bar{A}\bar{C}}}) && \text{De-Morgan's Law}\\
&= \bar{B}(A \oplus C) + B(\overline{(\bar{A} + \bar{C}) \cdot (A + C)}) && \text{De-Morgan's Law}\\
&= \bar{B}(A \oplus C) + B(\overline{\bar{A}A + \bar{A}C + A\bar{C} + \bar{C}C}) && \text{Distributive property}\\
&= \bar{B}(A \oplus C) + B(\overline{\bar{A}C + A\bar{C}}) && \text{Complement Law}\\
&= \bar{B}(A \oplus C) + B(\overline{A \oplus C}) && \text{Distributive property}\\
&= B \oplus (A \oplus C) && \text{Definition of XOR}\\
&= B \oplus A \oplus C && \text{Associative property}\\
&= A \oplus B \oplus C && \text{Commutative property}
\end{aligned}$$

Both of the functions are then implemented using AOI logic in Quartus.

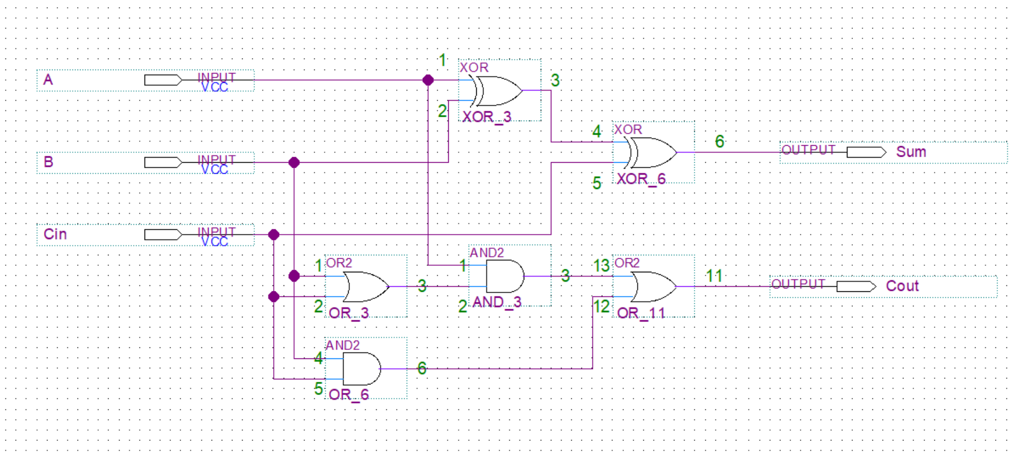


Figure 2: Circuit simulation using the test bench

Figure 2 is a circuit diagram of the three-input, four-output full adder using the boolean expressions previously derived.

4-bit Adder/Subtractor

Adding binary numbers can be achieved by chaining multiple full-adders together can wiring the C_{out} of one full-adder to the C_{in} of the next adder. Subtraction can be thought of as adding a negative number. When a number is represented in 2's complement, adding negative numbers works just as their positive counterparts. A 2's complement number can be formed from its opposite by negating each bit and adding 1 to the result. Therefore subtraction can be done by inverting the inputs to each of the bits of one number and then setting the C_{in} of the first adder to 1. By setting the carry to 1, we are effectively adding 1 to the summation of the first bit. However, merely adding an inverter would limit the circuit to only subtraction. The XOR gate solves this problem by acting as an inverter for the first input if the second input is 1 and a wire if the second input is 0. Therefore a control pin can be used to invert the inputs of one number and act as the C_{in} for 4-bit adder.

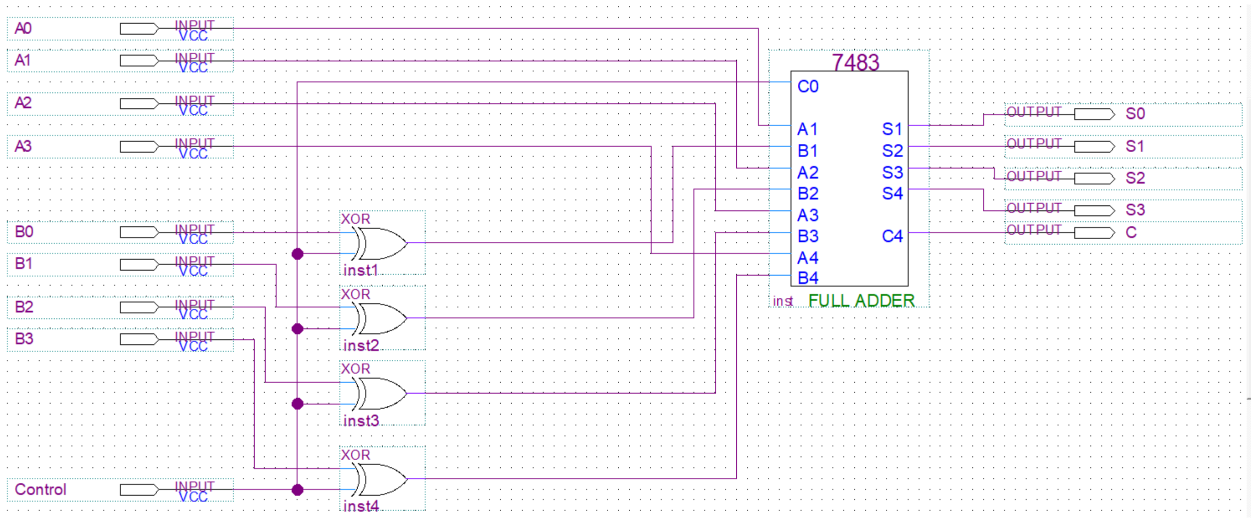


Figure 3: 4-bit full adder with subtraction control.

Figure 3 shows the Quartus diagram of the adder/subtractor. The two binary numbers A and B are represented with four input pins. $A0$ and $B0$ are the least significant bits (LSB). The inputs are feeding into a 74LS83 chip which consists of four fast-adders. The final input pin is labeled control. When set to `HIGH` the control pin will invert the signals for B. The control is also connected to $C0$ on the adder chip. This will add one as previously discussed.

The outputs $S0$ through $S4$ represent the bits on the output from the summation of the two input numbers. C is the carry from the final adder inside the 74LS83 chip.

Results and Analysis

1-bit full adder

The single full-adder designed in the previous part was tested inside Multisim to verify the outputs. A VHDL test bench was provided to control the inputs A, B, and C.

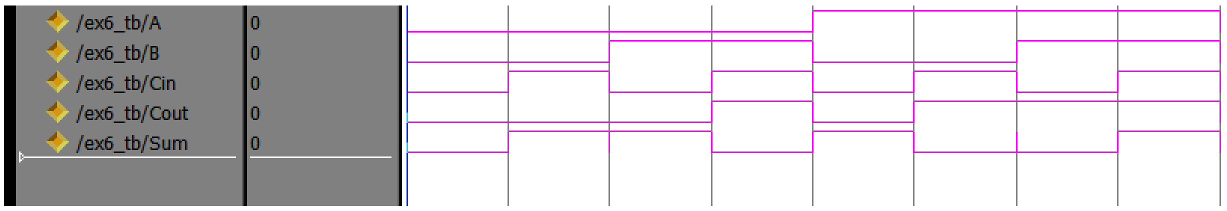


Figure 4: Simulation of full adder implementation.

Figure 4 shows the outputs of the sum and carry functions. The results match those in the Karnaugh Map in Figure 1. This means that the circuit was implemented correctly.

4-bit adder/subtractor

The circuit pictured in Figure 3 was constructed on a breadboard and then tested using a variety of input patterns.

Table 1: Four-bit binary addition

A3 A2 A1 A0	B3 B2 B1 B0	C4	F3 F2 F1 F0	$A + B = F$
1 0 1 0	0 0 1 1	0	1 1 0 1	$(-6) + (3) = -3$
1 1 1 1	1 1 1 1	1	1 1 1 0	$(-1) + (-1) = -2$
1 1 1 1	0 1 1 0	1	0 1 0 1	$(-1) + (6) = 5$
0 1 1 1	0 0 1 1	0	1 0 1 0	$(7) + (3) = -6$
0 1 0 1	1 0 1 0	0	1 1 1 1	$(-6) + (5) = -1$

Table 2: Four-bit binary subtraction

A3 A2 A1 A0	B3 B2 B1 B0	C4	F3 F2 F1 F0	$A - B = F$
1 0 1 0	0 0 1 1	1	0 1 1 1	$(-6) - (3) = 7$
1 1 1 1	1 1 1 1	1	0 0 0 0	$(-1) - (-1) = 0$
1 1 1 1	0 1 1 0	1	1 0 0 1	$(-1) - (6) = -7$
0 1 1 1	0 0 1 1	1	0 1 0 0	$(7) - (3) = 4$
0 1 0 1	1 0 1 0	0	1 0 1 1	$(5) - (-6) = -5$

Table 1 shows five examples of binary addition. All of the outputs correspond to those in decimal except for the fourth row. Here the output of $7 + 3$ turns out to be -6. This is because the 7 is the maximum positive number a 4-bit 2's comp number can hold. This is because the first bit is for the sign therefore there are only 3-bits that can hold the number.

When the addition overflows the number will wrap to the smallest negative value (-8). Rows 1 and 5 in Figure 2 run in to the same problem.

Conclusion

This exercise was implemented a full adder using simple logic gates. Then a 4-bit adder to create an adder/subtractor with 9-inputs and 5-outputs. A simulation of the full adder was performed in Multisim to verify the circuit against a Karnaugh Map. Both the full adder and the 4-bit adder/subtractor were constructed on a breadboard to test the signal outputs using LEDs. The exercise was successful in that the correct outputs were displayed given arbitrary inputs.

Questions

1. The simplest way to combine two full adders is to wire the C_{out} of the LSB to the C_{in} of the MSB.

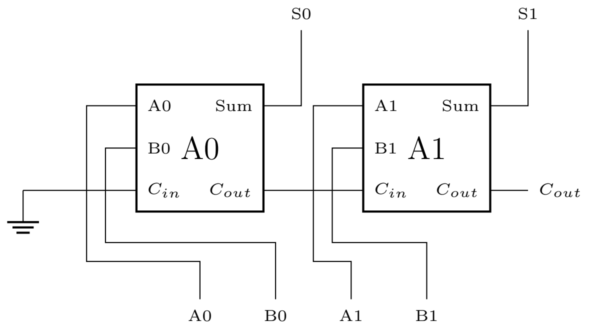


Figure 5: Two bit ripple-carry adder.

Figure 5 shows two full adders chained together by feeding the C_{out} of the first into the C_{in} of the second. The C_{in} of the LSB is connected to ground or 0 because there is no previous bit to carry from.

2. Overflow can be detected in an adder by analysing the signs of the inputs and output. If the signs of the inputs are opposite, there is no chance for overflow because the output is closer to zero relative to the inputs. An overflow occurs when the signs of the inputs are different from that of the output. For example in a 3-bit binary number, $1 + 7 = -8$. The inputs are both positive however the output is negative therefore an overflow occured. A boolean expression can be used to define this behavior. Given A_N and B_N being the N$^{\text{th}}$ bit in an N-bit binary number as inputs and F_N the N$^{\text{th}}$ bit in the output, the following expression can be derived to define $F_{overflow}$.

$$F_{overflow} = (AB + \overline{A_N}\,\overline{B_N}) \cdot (A_N \oplus F) = \overline{A_N \oplus B_N} \cdot (A_N \oplus F_N) \tag{3}$$

Equation 3 shows two implementations using a XNOR gate and an XOR gate or and XOR gate and the constituants of the XNOR. The first compontent checks if the signs of the inputs are the same, the second portion checks if that sign is different from the sign of the output. If both of those conditions meet, an overflow occured.

3. The circuit from Part 2 produces incorrect answers in some of the test cases because an overflow occured. Because the number is 2's comparison, the first bit acts as the sign and therefore a number may not exceed over 7 or under -8. When a sum of two numbers goes beyond the range described, the number will loop around to the other extreme.

4. If the result was extended to use a fifth bit, the answer would not be completely correct. The carry out of the four-bit adder needs to be added to the fifth bit of the augend and addend. However if the sign of the two operands are same, the result of the carry out bit can be used. To fix this, if the signs of the operands are different, the value of the fifth bit is the inverse of the carry out. If the operands are the same, the carry out may be used as the fifth bit. This can be defined using the following expression.

$$F_{N+1} = (A_N \oplus B_N) \oplus C_N \tag{4}$$

If the sign of A and B are different the first XOR gate will be 1. If this output is 1, the second XOR gate will act as an inverter. If they are equal, the second XOR will act as a wire.

5. Design a 4-bit adder/subtractor so that the overall circuit will produce a 5-bit result that is always correct in the case of overflow.

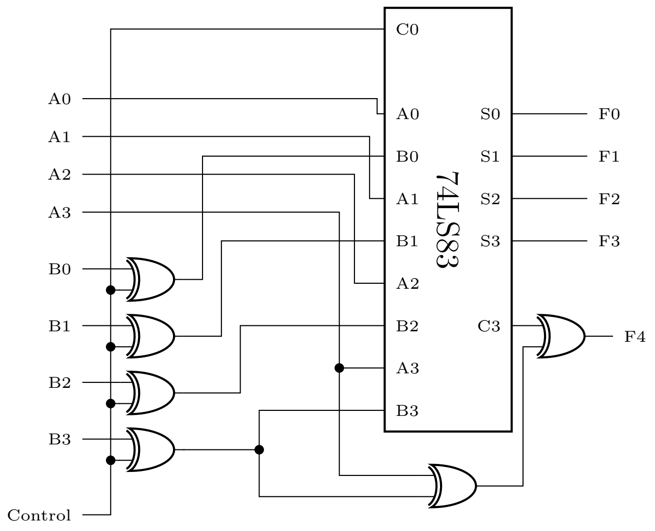


Figure 6: 4-bit adder/subtractor with 5-bit output.