CALIFORNIA STATE UNIVERSITY
LOS ANGELES

Department of Electrical and Computer Engineering
EE-2449 Digital Logic Lab

EXPERIMENT 8

ADDERS AND SUBTRACTORS

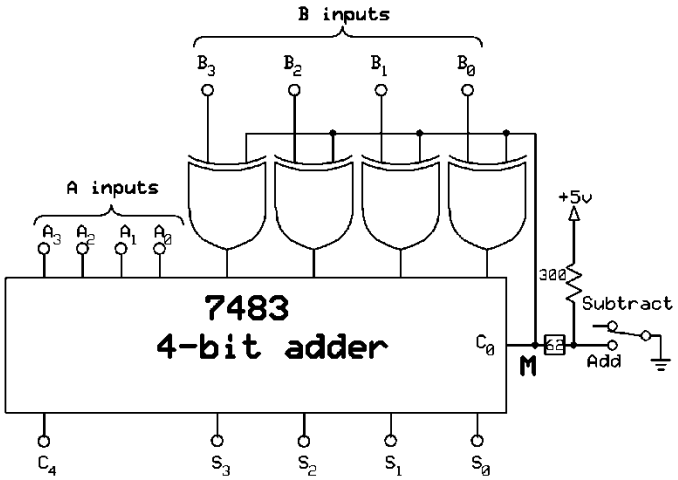
Text: Mano and Ciletti, *Digital Design, 5th Edition,* Chapter 2 Sec. 4-5.
Required chips: **7483:** Adder, plus some of the following:
7400: NAND **7402:** NOR **7404:** Inverter **7408:** AND **7432:** OR **7486:** XOR.

8.1 The 7483 is a 4-bit parallel adder. It consists of a chain (or "cascade") of 4 full-adder circuits. A subtractor equivalent of the 7483 could similarly be created by cascading four 1-bit full-subtractor circuits.

A full-subtractor has a truth table very much like that of a full adder. Its outputs are a difference bit and a borrow bit. With four of these chained together to produce a true 4-bit subtractor, you could generate the difference between two numbers, (A3..A0) and (B3..B0), as well as a borrow-out. If A < B, the borrow-out would be high.

Such a subtractor chip is not available, probably because it is easy enough to modify the 7483 so that it also subtracts. With this circuit, "subtraction" is done by 2's-complement addition.

A select input M (for Minus) is brought to the 7483-based adder/subtractor circuit from a toggle switch. It determines whether input B is added to or subtracted from input A. When M is 0, the circuit adds; when
M =1, it subtracts.



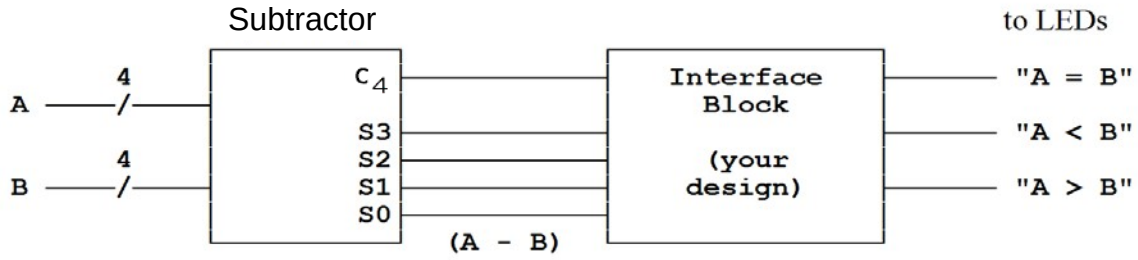
C_4 is the adder's carry-out bit. The 4-bit output is S = (S3..S0). It stands for sum when adding, but when subtracting, it represents the difference. In other words:
When M=0, output S = (A3..A0) + (B3..B0) + 0
When M=1, output S = (A3..A0) + (B3'..B0') + 1

The 0 and 1 at the right are from the 7483's carry input, Co, which is connected to M. In the second expression above, M complements the B inputs and adds the 1. The result is A + (B' + 1) = A - B. Complementing each Bn is done with an XOR controlled by M: if M=0, output = Bn; if M=1, output = Bn'.

8.2* Build the adder/subtractor described in 8.1. Let the A's come from one 4-switch block and the B's from another. First, test it by adding two pairs of 4-bit numbers. One pair should produce a sum that is within the range of an 4-bit unsigned number (sum ≤ 15). The other set should produce a sum that is outside the range of an 4-bit number. *Record the values used and the resulting outputs, including the carry-out.* In your lab notebook, compare your results with hand calculations using binary addition. Based on your observations, note what is the significance of the carry-out from the adder?

Then subtract using several different values of A and B. When performing subtraction, include all 3 cases: (1) A = B; (2) A < B; and (3) A > B. *Record the values used and the resulting outputs, including the carry-out.* In your lab notebook, compare your results with hand calculations using 2's-complement addition for subtract. Also, for each case, what is the relationship between the carry-out generated by this circuit with the borrow that would be generated by a true subtractor? You will need to know this for the next section.

8.3* The circuit of 8.2, configured as a subtractor, can be used to compare two 4-bit numbers. Subtractor outputs C_4 and (S3..S0) can be combined logically to indicate the relationship between inputs A and B. The gate circuit needed to do this appears as the interface block in the following diagram.



Design the contents of this interface. You will build the circuit using only **2** chips:

1. a **7402** NOR chip
2. *either* a **7432** OR chip or a **7408** AND chip.

Design the interface two ways: (a) an OR and NOR circuit, and (b), a NOR and AND circuit. Include ***both designs*** in your lab notebook and build and test one of them. Each design consists of 3 parts--one for each output--and requires 5 gates total. ExpressSCH has custom DM symbols for all gates (including ANDs and ORs). Use them, but **only** where they make circuit logic easy to follow (i.e. where bubbles cancel).

For each design:

(1) Start by drawing the circuit to generate "A = B". This output is true only when A - B = 0; i.e. only when (S3..S0) = 0000. Therefore, for this part of the interface block, you must combine inputs S3..S0 using gates so that output "A = B" goes high when S3..S0 are all low. A 4-input NOR would be the perfect solution--if one were available. Instead, you will have to do the design with the gates assigned.

(2) Next draw the circuit to generate output "A < B". This output is true if a borrow results when subtracting B from A. With a true subtractor, you would just connect "A < B" to the borrow bit. However, the "subtractor" used here is really a 2's-complement adder, so it produces a carry-out instead of a borrow. Therefore you must use the relationship between carry-out and borrow observed in Section 8.2

(3) Finally, draw the circuit for "A > B". This output goes high when the other two are low, since "A > B" is true only when "A = B" and "A < B" are both false.

Again: use DeMorgan gate symbols *wherever* they contribute to making circuit logic clear.

Implement one of your two designs (either one--your choice) in hardware. Connect it to the subtractor outputs and test it for these three cases:

A = B, A < B, and A > B. (You pick A and B.). Demonstrate your results for your instructor.

In your lab notebook, present a clear discussion of the following based on your lab results:

- If A = B, what are subtractor outputs S3..S0 and C_4? Explain why output “A = B” goes high.
- If A < B, what are S3..S0 and C_4? Explain why output “A < B” goes high.
- If A > B, what are S3..S0 and C_4? Explain why output “A > B” goes high.

Reminder: refer to manual section "Implementing and Troubleshooting Designs" in case you get stuck trying to debug your circuit.
