CS330 Architecture and Organization
Assignment Appendix B

Solution key

Problem B.2 — (5 points) In addition to the basic laws we discussed in this section, there are two important theorems, called DeMorgan's theorems:

$$\overline{A+B} = \overline{A} \cdot \overline{B} \text{ and } \overline{A \cdot B} = \overline{A} + \overline{B}$$

Prove that the two equations for E in the example starting on page B-7 are equivalent by using DeMorgan's theorems and the axioms shown on page B-7. (Section B.2)

Answer: We begin with one version of E and apply axioms and DeMorgan's until we achieve the other version.

$$\begin{aligned}
E &= ((A \cdot B) + (A \cdot C) + (B \cdot C)) \cdot (\overline{A \cdot B \cdot C}) \\
&= ((A \cdot B) + (A \cdot C) + (B \cdot C)) \cdot (\overline{A} + \overline{B} + \overline{C}) \quad \text{by two applications of DeMorgan's} \\
&= (A \cdot B) \cdot (\overline{A} + \overline{B} + \overline{C}) + (A \cdot C) \cdot (\overline{A} + \overline{B} + \overline{C}) + (B \cdot C) \cdot (\overline{A} + \overline{B} + \overline{C}) \quad \text{by distribution} \\
&= A \cdot B \cdot \overline{A} + A \cdot B \cdot \overline{B} + A \cdot B \cdot \overline{C} \\
&\quad + A \cdot C \cdot \overline{A} + A \cdot C \cdot \overline{B} + A \cdot C \cdot \overline{C} \\
&\quad + B \cdot C \cdot \overline{A} + B \cdot C \cdot \overline{B} + B \cdot C \cdot \overline{C} \quad \text{by distribution and associativity}
\end{aligned}$$

If we apply the commutativity and the inverse laws, every term that contains a letter and its complement has a zero in it.

$$\begin{aligned}
&= B \cdot 0 + A \cdot 0 + A \cdot B \cdot \overline{C} \\
&\quad + C \cdot 0 + A \cdot C \cdot \overline{B} + A \cdot 0 \\
&\quad + B \cdot C \cdot \overline{A} + C \cdot 0 + B \cdot 0
\end{aligned}$$

Now, we are basically finished.

$$\begin{aligned}
&= 0 + 0 + A \cdot B \cdot \overline{C} \\
&\quad + 0 + A \cdot C \cdot \overline{B} + 0 \\
&\quad + B \cdot C \cdot \overline{A} + 0 + 0 \quad \text{by the zero law} \\
&= (A \cdot B \cdot \overline{C}) + (A \cdot C \cdot \overline{B}) + (B \cdot C \cdot \overline{A}) \quad \text{by the identity law.}
\end{aligned}$$

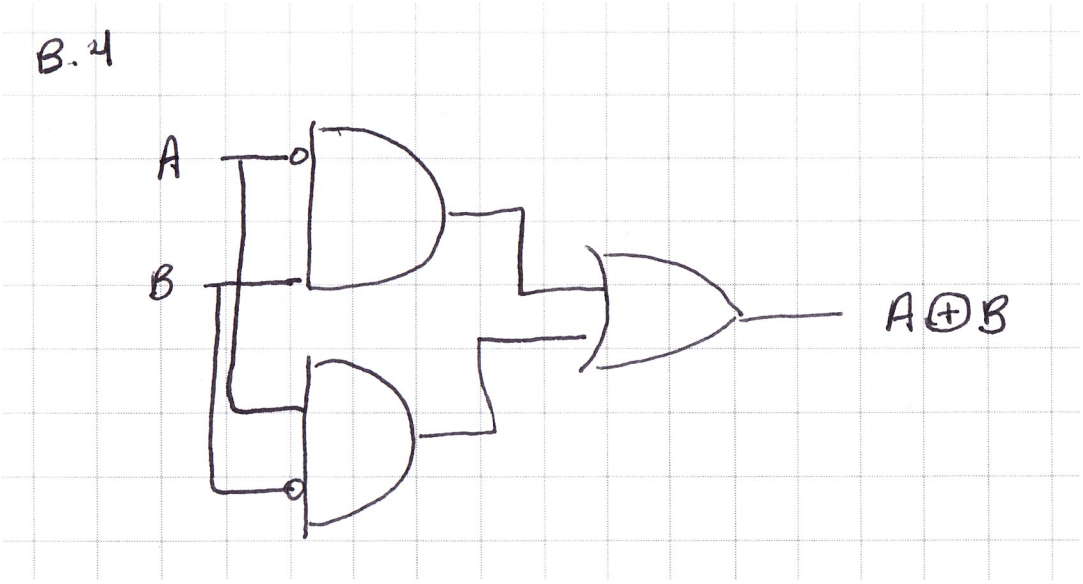
Problem B.4 — (6 points) One logic function that is used for a variety of purposes (including within adders and to compute parity) is *exclusive OR*. The output of a two-input exclusive OR function is true only if exactly one of the inputs is true. Show the truth table for an exclusive OR function and implement this function using AND gates, OR gates, and inverters. (Section B.4)

Answer:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

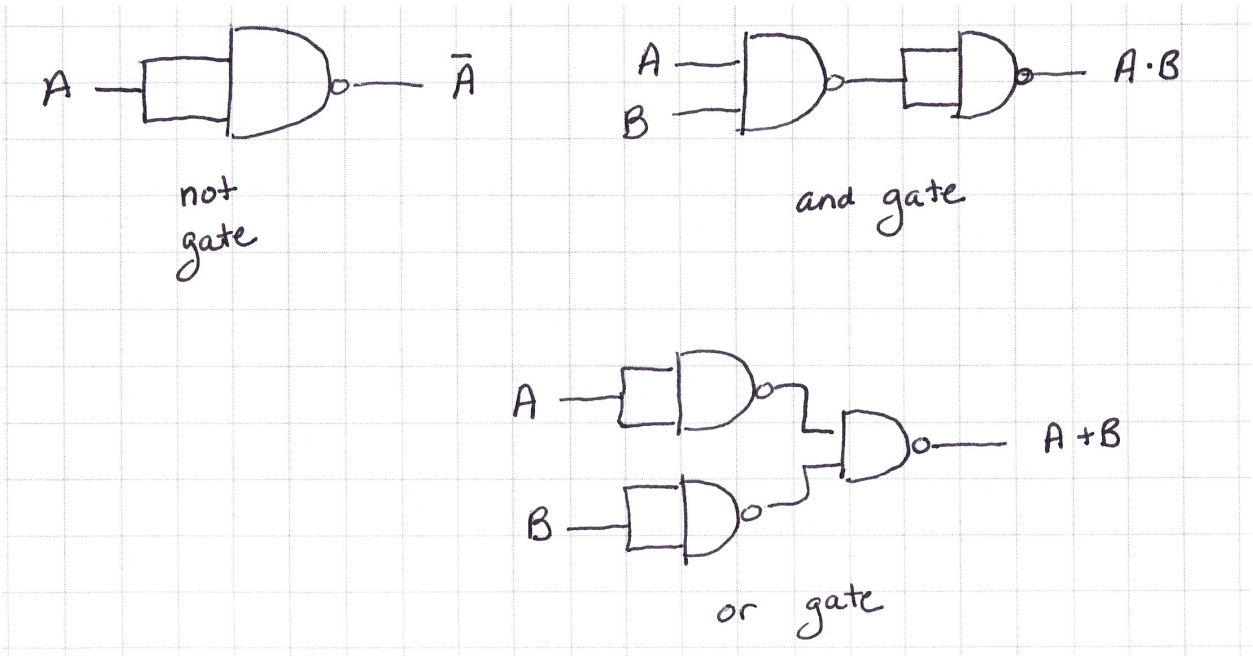
Algebraically, we can write

$$A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}.$$



Problem B.6 — (5 points) Prove that the NAND gate is universal by showing how to build the AND, OR, and NOT functions using two-input NAND gates. (Section B.2)

Answer:

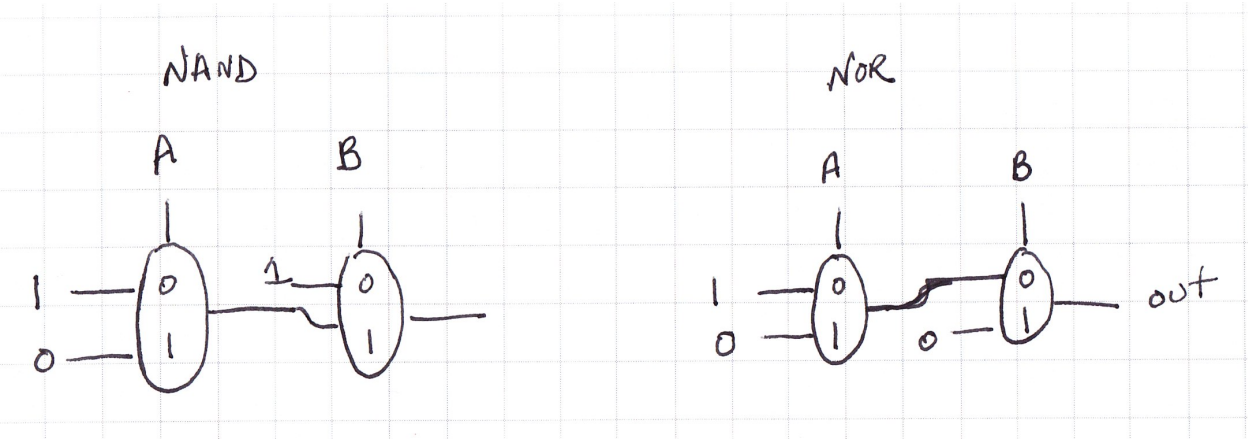


Problem B.10 — (5 points) Prove that a two-input multiplexor is also universal by showing how to build the NAND (or NOR) gate using two-input multiplexors. (Sections B.2 and B.3)

Answer: I found it helpful to make a truth table before attempting this.

A	B	$\overline{A \cdot B}$	$\overline{A + B}$
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

If we can configure two-input multiplexors to give the outputs in the NAND column or the NOR column, then we'll have shown it is universal.



Problem B.14 — (6 points) Implement a switching network that has two data inputs (A and B), two data outputs (C and D), an a control input (S). If S equals 1, the network is in pass-through mode and C should equal A and D should equal B. If S equals 0, the network is in crossing mode and C should equal B, and D should equal A. (Sections B.2 and B.3)

Answer: There are at least two ways to solve this problem. One is to realize that what is described could be considered a 3-input network, with inputs A, B, and S. Draw the truth table (which will have 8 rows) and then implement the truth table using gates.

S	A	B	C	D
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

Algebraically, we can write

$$C = \overline{S} \cdot \overline{A} \cdot B + \overline{S} \cdot A \cdot B + S \cdot A \cdot \overline{B} + S \cdot A \cdot B,$$

and

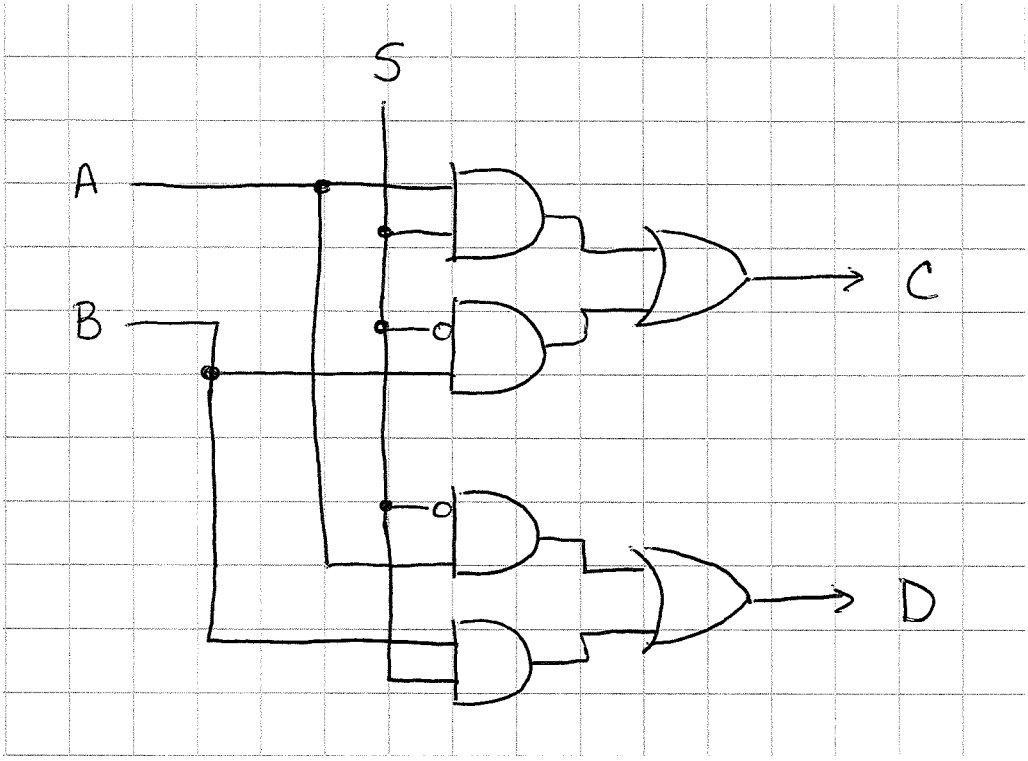
$$D = \overline{S} \cdot A \cdot \overline{B} + \overline{S} \cdot A \cdot B + S \cdot \overline{A} \cdot B + S \cdot A \cdot B.$$

These expressions may be drawn as gates, though it is somewhat complex. The astute student may notice that these can be simplified (either by manipulating the expressions using the axioms or by noticing that there are "don't care" values involved), so

$$C = \overline{S} \cdot B + S \cdot A,$$

and

$$D = \overline{S} \cdot A + S \cdot B.$$



Problem B.24 — (6 points) The ALU supported set on less than (`slt`) using just the sign bit of the adder. Let's try a set on less than operation using the values -7_{ten} and 6_{ten}. To make it simpler to follow the example, let's limit the binary representations to 4 bits 1001_{two} and 0110_{two}.

$$1001_{\text{two}} - 0110_{\text{two}} = 1001_{\text{two}} + 1010_{\text{two}} = 0011_{\text{two}}$$

This result would suggest that $-7 > 6$, which is clearly wrong. Hence, we must factor in overflow in the decision. Modify the 1-bit ALU in Figure B.5.10 on page B-33 to handle `slt` correctly. Make your changes on a photocopy of this figure to save time. (Section B.5)

Answer: The key observation is that the Overflow flag tells us when bit b_{31} incorrectly indicates the sign of the operation. So rather than copying b_{31} to the Set line, we want to use a truth table that incorporates the Overflow flag.

Overflow	b_{31}	Set
0	0	0
0	1	1
1	0	1
1	1	0

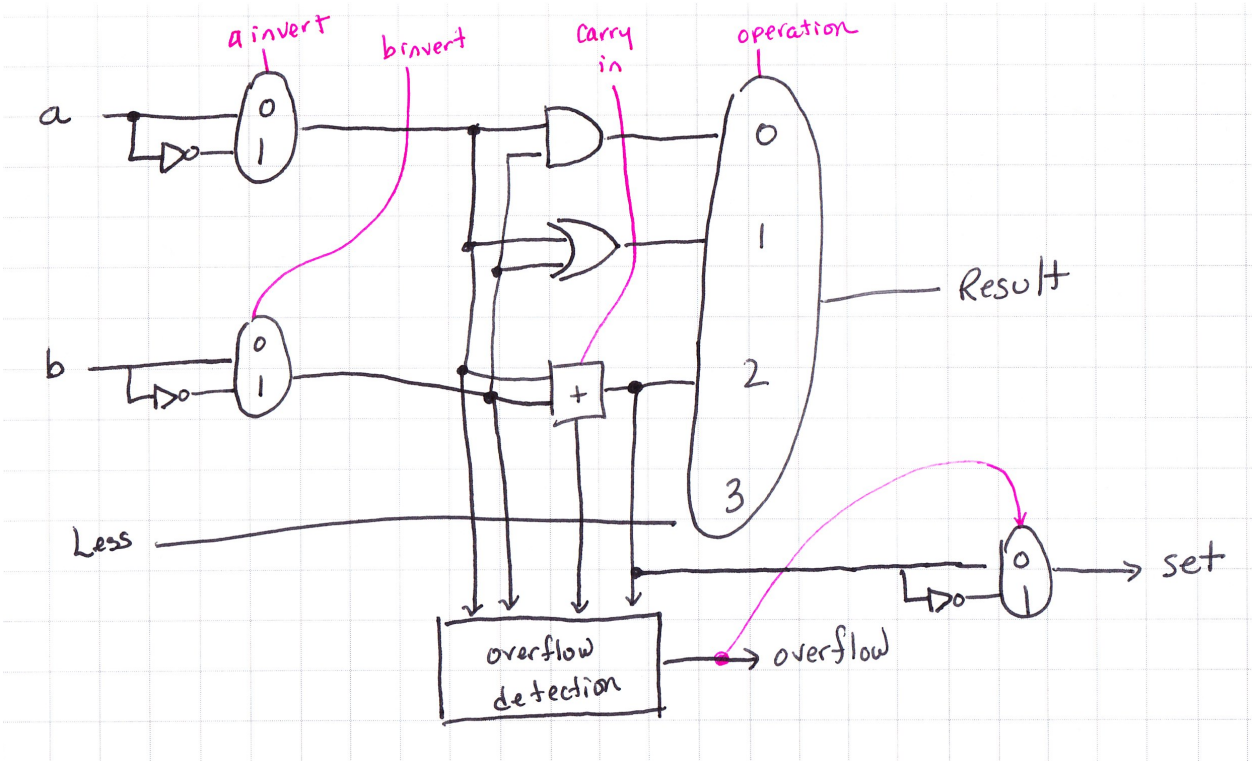
You may recognize this as exclusive OR, that is

$$\text{Set} = \text{Overflow} \oplus b_{31}.$$

Probably, it was assumed that we'd use AND, OR, and NOT gates to modify the figure, so we could write the expression using them instead:

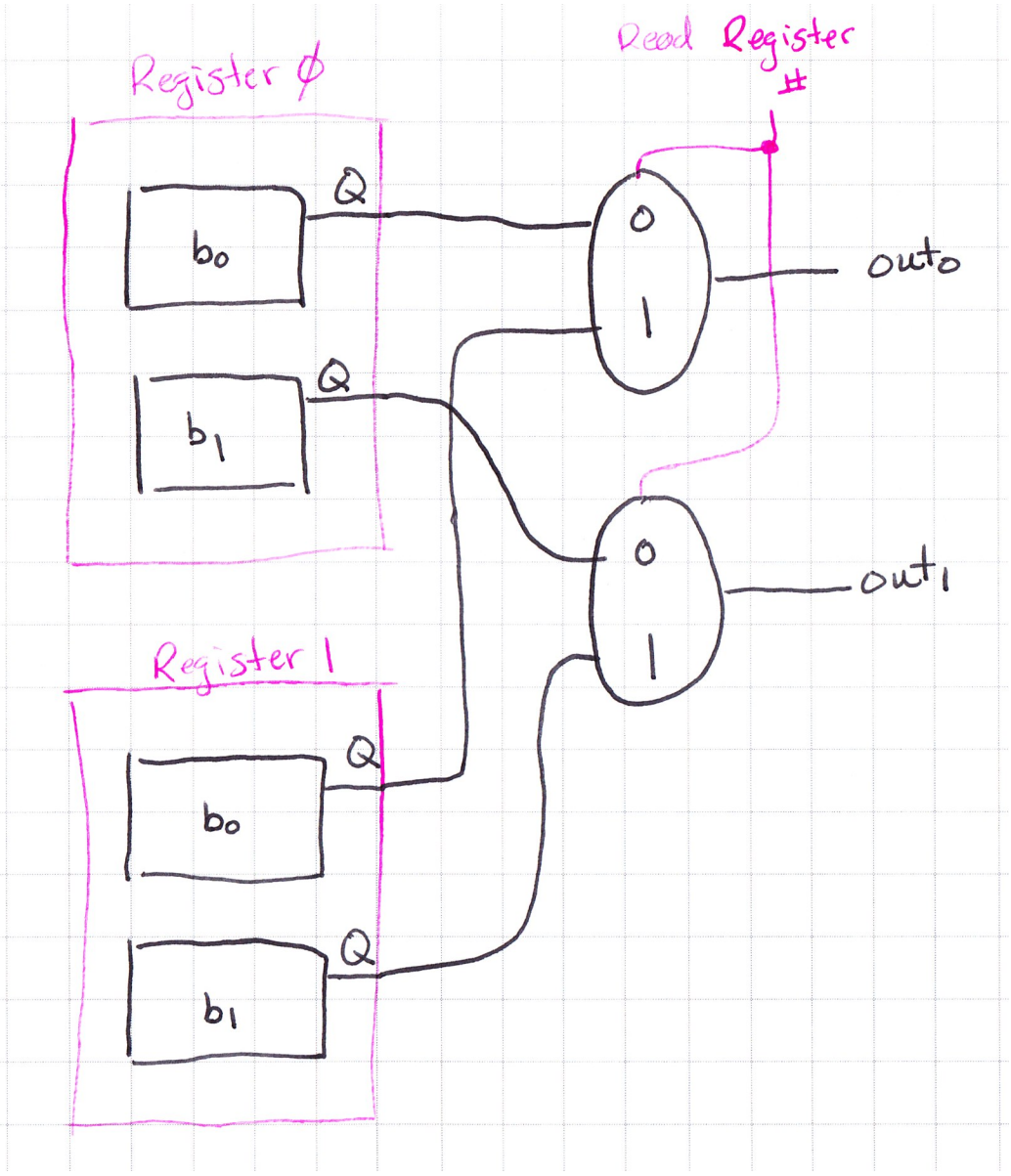
$$\text{Set} = \overline{\text{Overflow}} \cdot b_{31} + \text{Overflow} \cdot \overline{b_{31}}.$$

Another option would be to use a MUX fed with both b_{31} and $\overline{b_{31}}$ that is selected by the Overflow line. That is the figure I chose to draw.



Problem B.36 — (10 points) Figure B.8.8 on page B-55 illustrates the implementation of the register file for the MIPS datapath. Pretend that a new register file is to be built, but that there are only two registers and only one read port, and that each register has only 2 bits of data. Redraw Figure B.8.8 so that every wire in your diagram corresponds to only 1 bit of data (unlike the diagram in figure B.8.8, in which some wires are 5 bits and some wires are 32 bits). Redraw the registers using D flip-flops. You do not need to show how to implement a D flip-flop or a multiplexor. (Section B.8)

Answer:


Register 0
Read Register #
Q
b_0
Q
b_1
0
1
out_0
0
1
out_1
Register 1
Q
b_0
Q
b_1