Answers Homework 5

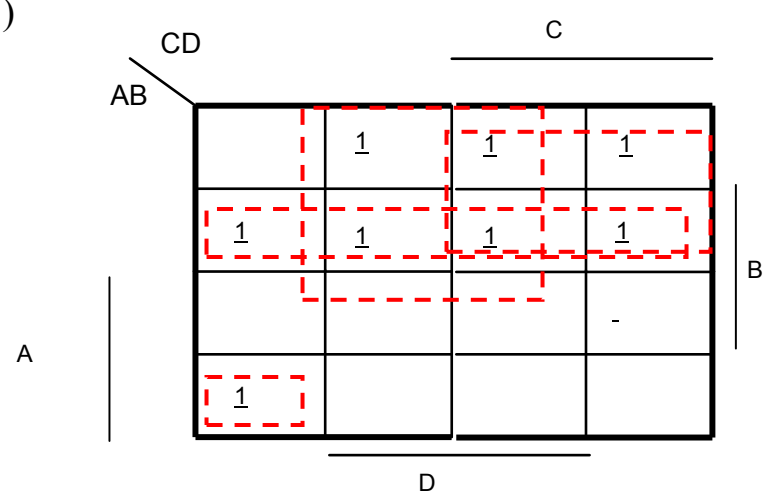
<u>Task 1)</u> Design two versions (Version A and Version B below) of the combinational circuit whose input is a 4-bit number and whose output is the 2's complement of the input number:
Version A) The circuit is a simplified two-level circuit, plus inverters as needed for the input variables.

We first set up the truth table for conversion to 2's complement so that we can minimize the functions for each output with K-maps (we have 4 outputs in total). To find the 2's complement of a binary number N, just flip the bits of N and add 1 (you should know this from lecture 1).

Input 4-bit number				2's complement of the input number			
A	**B**	**C**	**D**	**S1**	**S2**	**S3**	**S4**
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

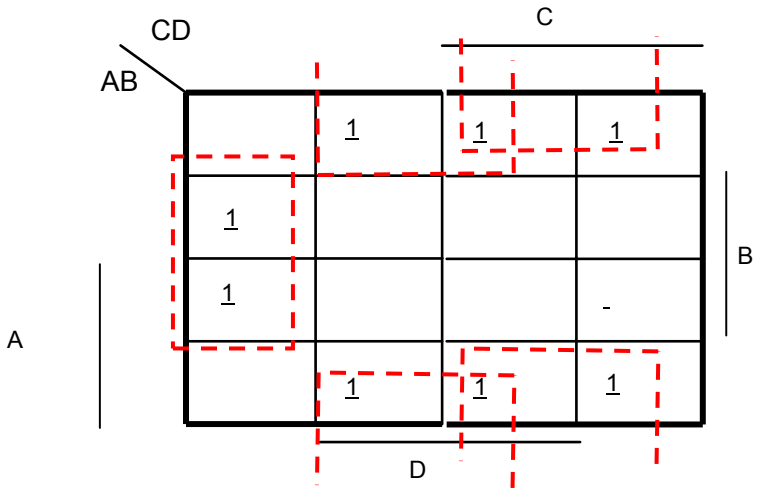
Now we can minimize the functions S1, S2, S3, and S4. Each output bit Si has its own K-map:

S1)



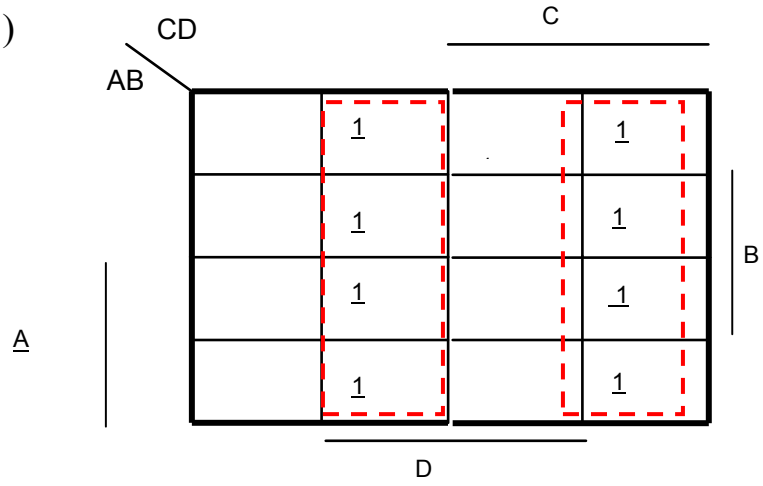
S1 = A'B + A'C + A'D + AB'C'D'

S2)



S2 = B'C + B'D + BC'D'

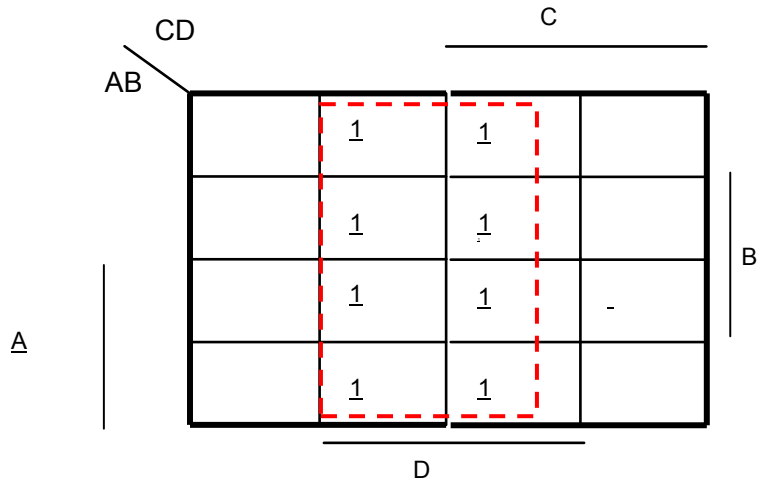
S3)



(notice that this is the exclusive or between C and D)

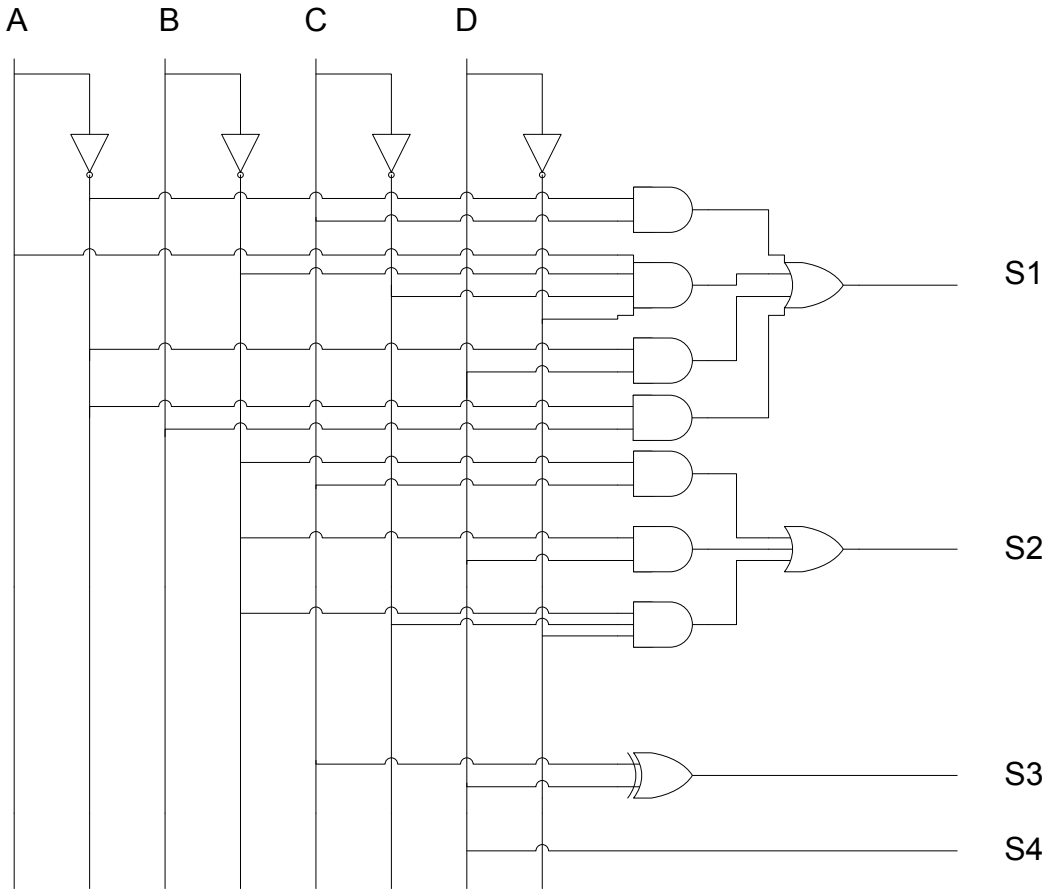
S3 = C'D + D'C = C xor D

S4)



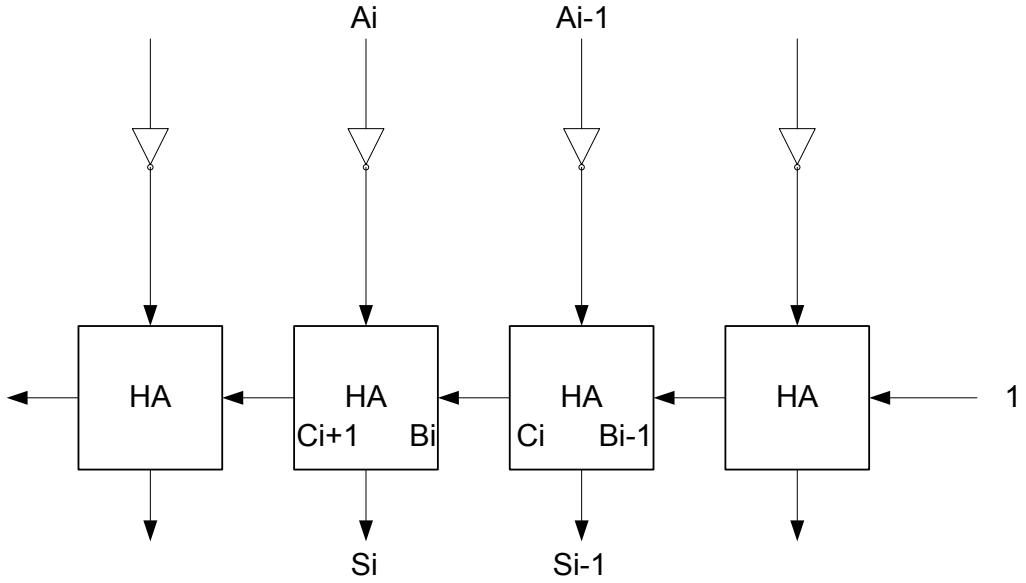
S4 = D

Now, we have all minimized functions and we can draw our circuit.



Version B) The circuit is made up of four identical two-input, two-output cells, one for each bit. The cells are connected in cascade, with lines similar to a carry between them. The value applied to the rightmost carry bit is 0.

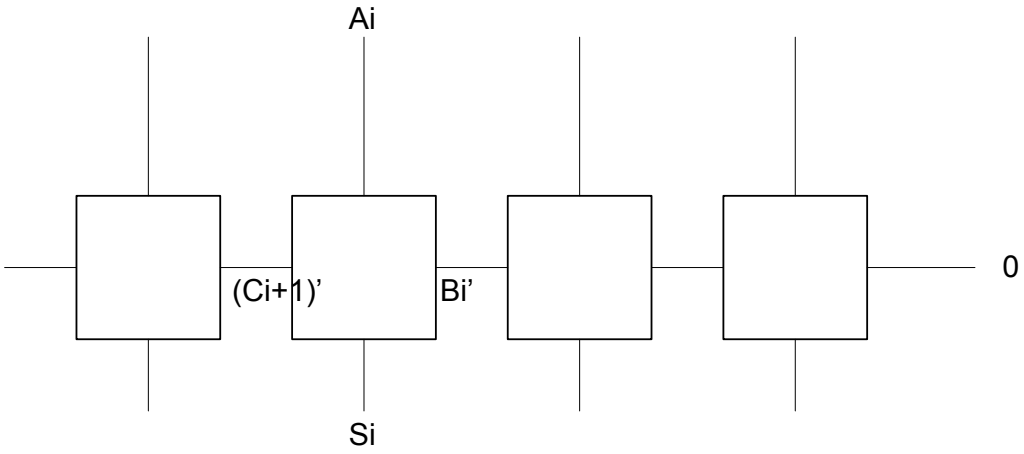
Here's an almost identical solution that uses 4 half adders and a rightmost carry of 1 (see the lectures). We want the rightmost carry to be 0 and to remove the inverters. So, we have to modify this circuit. Let us take the third block and make the Boolean equations for the 2 outputs Si, Ci+1.



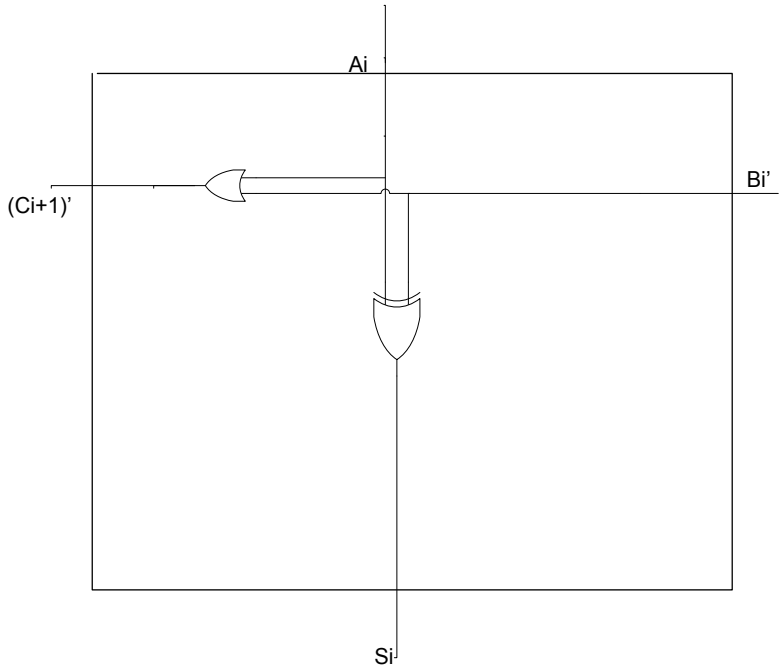
We can show that **Si is Ai xor Bi'**. This makes our work a lot easier.
Si = Ai' xor Bi = Ai'Bi' + AiBi = (See Lecture 5 for half adders)
= Ai Bi + Ai'Bi' = **Ai xor Bi'**

If we want to use Bi' then we have to use (Ci)' from the previous block and to generate (Ci+1)' for the next block
Ci+1 = Ai' Bi
(Ci+1)' = (Ai' Bi)' = **Ai + Bi'**

Now we have the needed Boolean equations to make our circuit. Since, we complemented Bi we can use 0 instead of 1 in the right most carry (all blocks are identical).

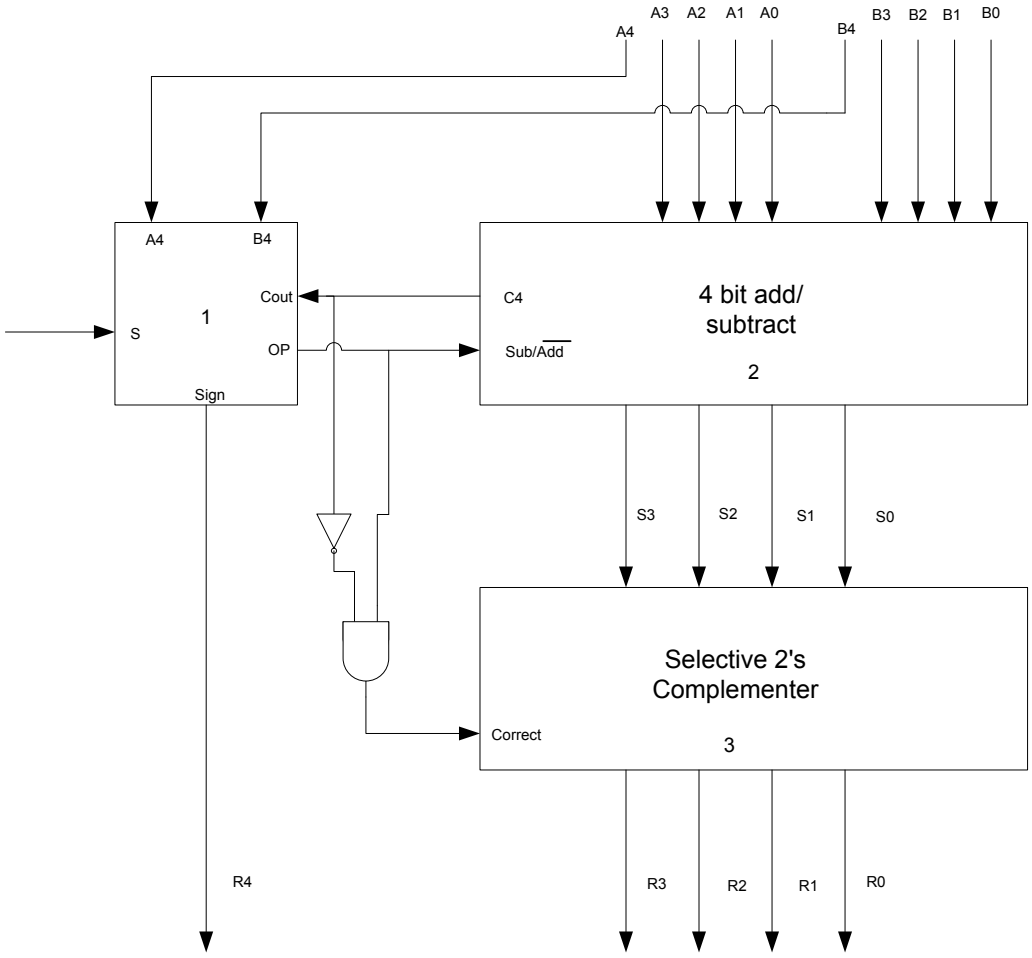


Below, we draw the circuit that is inside of one of the blocks (all blocks are identical). The only difference with a half adder is the OR gate (a half adder uses an AND gate)



<u>Task 2)</u> Design a 5-bit signed magnitude adder-subtractor (1 bit for the sign and 4 bits for the magnitude). Divide the circuit for design into: (1) sign generation and add/subtract control logic, (2) an unsigned number adder-subtractor using 2's complement of the subtrahend for subtraction, and (3) selective 2's complement result correction logic.

Here is the circuit divided in three parts. (1) add/subtract control logic, (2) an unsigned adder-subtractor and (3) a selective 2's complement result correction logic. Let us examine the individual parts more closely.

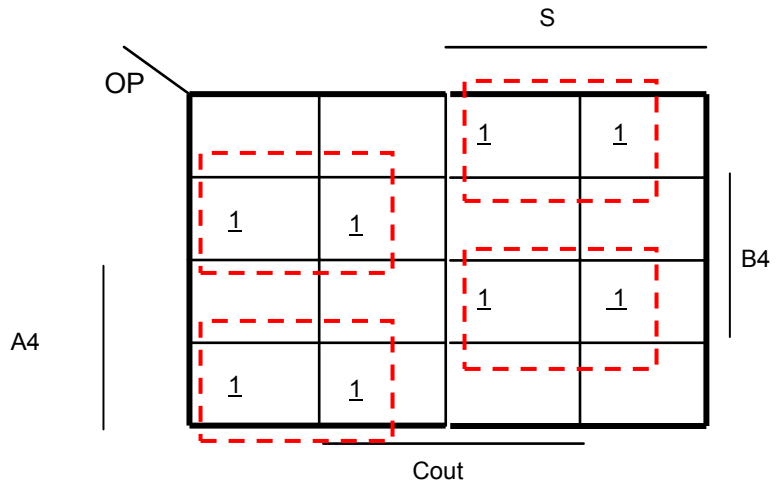


Circuit 1) **S** is the operation we want to do (S = 1 is subtraction, S = 0 is addition). **A4** and **B4** are the sign bits of the signed numbers. **Cout** is the carry out from circuit (2). **OP** is the signal that determines the real operation (add or subtract) depending on the signs A4 and B4 as well as from the input S. After we have checked the signs and carry we can determine the final sign (**R4**) for the addition/subtraction.

Let us set up a truth table for this purpose. The outputs are **OP** and **Sign**, the rest is all input.

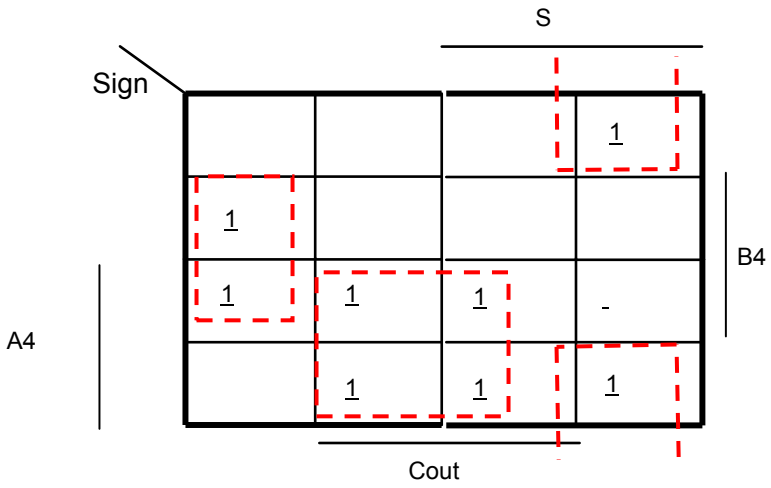
A4	B4	S	Cout	OP	Sign	meaning
0	0	0	0	0	0	(+A) + (+B), +(A+B)
0	0	0	1	0	0	overflow
0	0	1	0	1	1	(+A) – (+B), B>A, -(A-B)
0	**0**	**1**	**1**	**1**	**0**	(+A) – (+B), A>B, +(A-B)
0	1	0	0	1	1	(+A) + (-B), B>A, -(A-B)
0	1	0	1	1	0	(+A) + (-B), A>B, +(A-B)
0	1	1	0	0	0	(+A) - (-B), B>A, +(A+B)
0	1	1	1	0	0	(+A) - (-B), B>A, +(A+B)
1	0	0	0	1	0	(-A) + (+B), B>A, +(A-B)
1	0	0	1	1	1	(-A) + (+B), A>B, -(A-B)
1	0	1	0	0	1	.. you get the point
1	0	1	1	0	1	.. you get the point
1	1	0	0	0	1	.. you get the point
1	1	0	1	0	1	.. you get the point
1	1	1	0	1	0	.. you get the point
1	1	1	1	1	1	.. you get the point

K-Map for **OP**



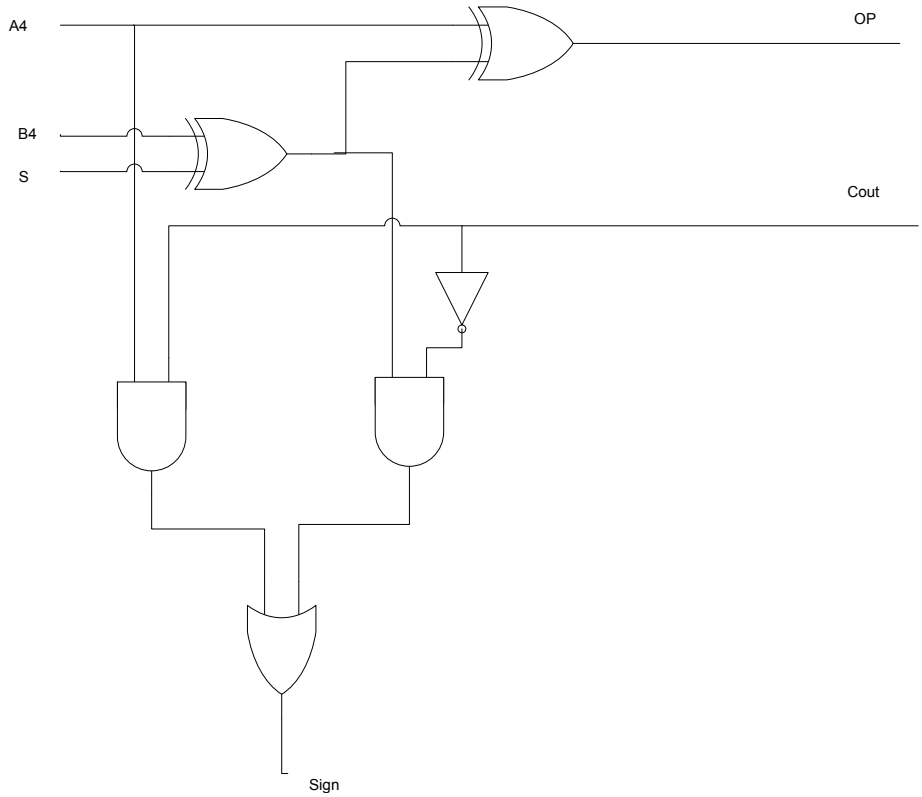
OP = A4 B4’ S’ + A4 B4 S + A’ B4 S’ + A4’ B4’ S = **A4 xor B4 xor S**

K-Map for **Sign**



Sign = A4 Cout + B4 S' Cout' + B4' S Cout'
= **A4 Cout + Cout' (B4 xor S)**

Now we can draw the circuit.



<u>Circuit 2)</u> See Lecture 5 for the circuit of unsigned 2's complement Adder-Subtractor.

Circuit 3) Selective 2's complementer. We only want to take the 2's complement when needed. (See the truth table for (1) if you want to know why) so we use the correct line which is 1 if the carry is zero and a subtract operation is used. If correct is 1 it complements and adds 1 to the input, else it does nothing and the output equals the input.