ALU Functions:

1. ADD:

   Implemented by using the adder given. inv_r and in_s are both set to high because they are active low. We want to add two positive numbers.

2. SUBR

   Uses the adder from ADD but we have inv_r = 0 and inv_s = 1 and C_{in} = 1, giving us S + (-R), where (-R) is just the two's complement of R.

3. SUBS

   Uses the adder from ADD but we have inv_r = 0 and inv_s = 1 and C_{in} = 1, giving us S + (-R), where (-R) is just the two's complement of R.

4. OR

   We use the *logic* in our ALU to get the OR value. We have inv_r = 0 and inv_s = 0. We use DeMorgan's law to get R OR S from ~R NAND ~S.

5. AND

   We use the *logic* in our ALU to get the AND value. We have inv_r = 0 and inv_s = 0. We use DeMorgan's law to get R AND S from ~R NOR ~S.

6. NOTRS

   We use the same exact logic as the AND function but we set inv_r = 1 and keep inv_s = 0. This gives us R NOTRS S from R NOR ~S

7. XOR

   We use the XOR gate in the ALU logic to produce the XOR function. We set inv_r = 1 and inv_s = 1 to make sure our logic uses R and S.

8. XNOR

The XNOR function uses the exact same signals as XOR but we invert the output to get XNOR.