Sign-Magnitude Addition

Problem Statement

Sign-magnitude addition is harder than two's complement addition. Consider adding N-bit numbers Y = A + B. Let SY, SA, and SB be the sign bits and MY, MA, and MB be the N-1 bit magnitudes.

An algorithm for sign-magnitude addition (disregarding overflow) is:

```
if (SA == SB) begin // if the signs are the same
    {Cout, MY} = MA + MB     // just add the magnitudes
    SY = SA          // and the sign of the result matches the sign of the
inputs
end else begin       // if the signs are different
    MBN = ~MB
    {Cout, MR} = MA + MBN + 1 // Negate MB using 2's complement, and then add A
+ (-B)
    if (Cout) begin // If there is a carry out, MA >= MB
MY = MR      // Result has correct magnitude
        SY = SA      // And sign matches A
    end else begin
        MY = ~MR + 1 // Otherwise negate result with 2's complement
        SY = SB      // and sign matches B
    end
end
```

Component delays are given below. Remember that the assign statements at the beginning and end of small just imply wires and cost no area or delay. Note that the design described in the SystemVerilog does not use NANDs or NORs.

Cell	t_{pd} (ps)	t_{cd} (ps)	Relative Size
NOT	6	4	2
AND2	14	10	5
AND3	18	14	6
OR2	16	12	5
OR3	20	16	6
XOR2	24	18	9
NAND2	8	6	3
NAND3	12	10	4
NOR2	10	8	3
NOR3	14	12	4

What is the critical path through the sign/magnitude adder? What is the critical path through the sign/magnitude adder? Suppose you were building the system with CMOS gates and could redesign circuits using NAND and NOR instead of AND and OR. Which module would you redesign to obtain the greatest speedup?