1. Short adders

In this question we are going to compare the speed of different types of adders.

a) Calculate the worst case delay of a Ripple carry adder consisting of the full adder blocks shown below. You can use $t_p$ for the AND and OR functions and $2t_p$ for the XOR gates. Express your answer in terms of $t_p$.

![Ripple carry adder diagram]

b) For the second part of this question you are to implement an 8-bit Carry-Look Ahead adder. For $n$ input AND/OR gates use the $t_{p-n} = 0.25*n^2*t_p$, similarly for the XOR gate $t_{XOR-n} = 0.5*n^2*t_p$. Find the worst-case delay of this adder again in terms of $t_p$.

c) Repeat the same calculations for 32-bit adders. Hint: Implement the Carry-Look Ahead Adder as a block CLA of 4-bit block-length.

2. Comparator design

For this problem you are given a cell library consisting of full adders and arbitrary fan-in logic gates (ie, AND, OR, INVERTER, etc.).

a) In the first part, our goal is to design an $N$-bit comparator with output $A > B$ using the library. If you use and adder do the implementation of the adder as a RIPPLE carry adder.

b) Then using the same gate delay values in the first problem estimate the delay of your comparator. If you can’t find anything better than using an adder use an ripple carry adder.
3. **Signed Multiplication**

In the lecture you were provided with a multiplication algorithm for unsigned numbers. Contrary to common sense, this algorithm does not extend to signed numbers by simply sign extending. Consider the example

\[ \begin{array}{c}
1 & 1 & 0 & 0 & \rightarrow -4 \\
\times & 1 & 0 & 0 & 1 & \rightarrow -7 \\
\hline
1 & 1 & 1 & 1 & 1 & 0 & 0 \\
+ & 1 & 1 & 0 & 0 \\
\hline
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & \rightarrow -36
\end{array} \]

But many times in real life you need to implement signed multipliers. In fact, the array multiplier given in figure 7.28 of your textbook can be modified to implement a signed multiplier. Your task is to perform these necessary modifications to convert the 4x4 multiplier of figure 7.28 into a signed multiplier. (Hint: the decimal value of a 2’s complement binary number

\[ b_{n-1} b_{n-2} \ldots b_1 b_0 = -b_{n-1} 2^{n-1} + b_{n-2} 2^{n-2} \ldots b_1 2^1 + b_0 2^0 \]

That is the MSB is attributed a negative value.