Constructing Carry Lookahead Adders

This assignment has two parts: The first part is to build a 4-bit carry lookahead adder, the second part is to construct 16-bit carry lookahead adder from four of your 4-bit adders.

Part 1: A 4-bit carry lookahead adder

As described in the book and in the lecture notes, a 4-bit carry lookahead adder has the following inputs and outputs:

**Inputs**

- A₀ – A₃ : 4-bit input value (input lines 0-3)
- B₀ – B₃ : 4-bit input value (input lines 4-7)
- c₀ : 1-bit carry-in (input line 8)

**Outputs**

- R₀ – R₄ : 4-bit output value (output lines 0-3)
- cₒᵤₜ : 1-bit carry-out (output line 4)
- P : 1-bit “super propagate” value (output line 5)
- G : 1-bit “super generate” value (output line 6)

In order to determine all the carry’s for all four bits at once, the 4-bit carry lookahead adder computes the following quantities:

\[
\begin{align*}
  p_i &= A_i + B_i \quad \text{for } i \in \{0..3\} \\
  g_i &= A_i \cdot B_i \quad \text{for } i \in \{0..3\} \\
  c_1 &= g_0 + (p_0 \cdot c_0) \\
  c_2 &= g_1 + (p_1 \cdot g_0) + (p_1 \cdot p_0 \cdot c_0) \\
  c_3 &= g_2 + (p_2 \cdot g_1) + (p_2 \cdot p_1 \cdot g_0) + (p_2 \cdot p_1 \cdot p_0 \cdot c_0) \\
  c_4 &= g_3 + (p_3 \cdot g_2) + (p_3 \cdot p_2 \cdot g_1) + (p_3 \cdot p_2 \cdot p_1 \cdot g_0) + (p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0)
\end{align*}
\]

where \( c_4 \) is the carry-out of the 4-bit adder. The values of \( A_i, B_i, \) and \( c_i \) can then be fed to the logic computing the result of each one-bit addition (remember, carry’s were already computed, so no carry-out from each 1-bit addition is necessary). You can simply re-use the “result” part of the code from your 1-bit adder constructed for the previous assignment.
The super propagate and super generate values, P and G, respectively, are used to determine the carry’s between the 4-bit adders when they are put together to form a 16-bit adder. P and G are computed as follows:

\[ P = p_3 \cdot p_2 \cdot p_1 \cdot p_0 \]
\[ G = g_3 + (p_3 \cdot g_2) + (p_3 \cdot p_2 \cdot g_1) + (p_3 \cdot p_2 \cdot p_1 \cdot g_0) \]

Note the similarity between G and \( c_4 \), above (although they are not identical). This means that you should be able to share the logic computing those portions of G and \( c_4 \) that are the same. Try to exploit similarities in other portions of the adder as well, to reduce the number of gates.

**Part 2: A 16-bit carry lookahead adder**

A 16-bit carry lookahead adder has the following inputs and outputs:

**Inputs**
- \( A_0 - A_{15} \): 16-bit input value (input lines 0-15)
- \( B_0 - B_{15} \): 16-bit input value (input lines 16-31)
- \( C_0 \): 1-bit carry-in (input line 32)

**Outputs**
- \( R_0 - R_{15} \): 16-bit output value (output lines 0-15)
- \( C_{out} \): 1-bit carry-out (output line 16)

The logic for creating a 16-bit carry lookahead adder from four 4-bit lookahead adders is quite straightforward. As described in class, some additional logic is required to compute, ahead of time, the carry-in for each of the 4-bit adders. This logic would implement:

\[ C_1 = G_0 + (P_0 \cdot C_0) \]
\[ C_2 = G_1 + (P_1 \cdot G_0) + (P_1 \cdot P_0 \cdot C_0) \]
\[ C_3 = G_2 + (P_2 \cdot G_1) + (P_2 \cdot P_1 \cdot G_0) + (P_2 \cdot P_1 \cdot P_0 \cdot C_0) \]
\[ C_4 = G_3 + (P_3 \cdot G_2) + (P_3 \cdot P_2 \cdot G_1) + (P_3 \cdot P_2 \cdot P_1 \cdot G_0) + (P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0) \]

where \( P_i \) and \( G_i \) refer to the P and G outputs, respectively, of the \( i^{th} \) 4-bit adder and \( C_4 \) would be the carry-out of the entire 16-bit adder. Each \( C_i \) is sent as the carry-in to the \( i^{th} \) 4-bit adder.

**What to turn in**

As in the previous assignment, your code should be put in the file `mycode.c` and emailed to the grader. The file should contain (at least) the definition of two procedures:

```
DEVICE new_4bit_lookahead_adder();
DEVICE new_16bit_lookahead_adder();
```
which return 4-bit and 16-bit carry lookahead adders. Naturally, 
new_16bit_lookahead_adder() should call 
new_4bit_lookahead_adder() four times to create the four 4-bit lookahead
adders used in a 16-bit lookahead adder. Be sure to write test code for the two kinds of
devices you build, to make sure that they work correctly.