For the delay we examine two possibilities:

- **Option 1:**
  \[
  t_{PLH}(x_0 \rightarrow c_4) = t_{PLH}(\text{XOR}2) + t_{PHL}(\text{NAND}5) + t_{PLH}(\text{NAND}5)
  \]
  \[
  t_{PLH}(x_0 \rightarrow c_4) = 0.3 + 0.036 \times 5.1 + 0.34 + 0.019 + 0.21 + 0.038L = 1.05 + 0.038L
  \]
  \[
  t_{PHL}(x_0 \rightarrow c_4) = t_{PHL}(\text{XOR}2) + t_{PLH}(\text{NAND}5) + t_{PHL}(\text{NAND}5)
  \]
  \[
  t_{PHL}(x_0 \rightarrow c_4) = 0.3 + 0.021 \times 5.1 + 0.21 + 0.038 + 0.34 + 0.019L = 1.00 + 0.019L
  \]

- **Option 2:**
  \[
  t_{PLH}(x_0 \rightarrow z_3) = \max\{t_{PLH}(\text{XOR}2) + t_{PHL}(\text{NAND}4) + t_{PLH}(\text{NAND}4), t_{PHL}(\text{XOR}2) + t_{PHL}(\text{NAND}4) + t_{PHL}(\text{NAND}4)\} + 0.16 + 0.036L
  \]
  \[
  t_{PLH}(x_0 \rightarrow z_3) = \max\{0.3 + 0.036 \times 5.1 + 0.12 + 0.051 + 0.10 + 0.037 \times 2, 0.3 + 0.021 \times 5.1 + 0.10 + 0.037 + 0.12 + 0.051 \times 2\} + 0.16 + 0.036L
  \]
  \[
  t_{PHL}(x_0 \rightarrow z_3) = \max\{0.66, 0.77\} + 0.16 + 0.036L
  \]
  \[
  t_{PHL}(x_0 \rightarrow z_3) = 0.93 + 0.036L
  \]
  \[
  t_{PHL}(x_0 \rightarrow z_3) = \max\{\ldots\} + 0.15 + 0.020L
  \]
  \[
  t_{PHL}(x_0 \rightarrow z_3) = 0.77 + 0.15 + 0.020L = 0.92 + 0.020L
  \]

From these equations we determine the network delay taking the largest values for LH and HL transitions from both options:

\[
 t_{PLH}(x_0 \rightarrow c_4) = 1.05 + 0.038L
 t_{PHL}(x_0 \rightarrow c_4) = 1.00 + 0.019L
\]

From these figures we conclude that the CLA adder uses more gates than the ripple carry adder but has a smaller propagation delay.

**Exercise 10.3** The BCD to Excess-3 converter using 4-bit binary adder is shown in Figure 10.1.

**Exercise 10.4** BCD Addition

When we add two BCD digits (0..9), considering a carry-in bit, the range of values obtained is from 0 to 19. The output consists of a carry out and a digit coded in BCD also.

\[
 s = (a + b + C_{IN}) \mod 10
 C_{OUT} = \begin{cases} 1 & \text{if } (a + b + C_{IN}) \geq 10 \\ 0 & \text{if otherwise} \end{cases}
\]

where \( s, a \) and \( b \) are BCD digits.

When the integers \( a \) and \( b \) are applied to the inputs of the binary adder, the output is:

\[
 z = (a + b + C_{IN}) \mod 2^4
 C_0 = \begin{cases} 1 & \text{if } (a + b + C_{IN}) \geq 2^4 \\ 0 & \text{if otherwise} \end{cases}
\]

So, looking at the binary adder output and comparing to the expected output for the BCD adder, we must consider three cases:
Figure 10.1: BCD to Excess-3 converter - Exercise 10.3

(i) \( C_0 = 0 \) and \( z < 10 \): the output of the binary adder does not need correction

(ii) \( 10 \leq z \leq 15 \) and \( C_0 = 0 \): in this case we convert the sum as follows:

\[
s = z \mod 10 = (z - 10) = (z + 6) \mod 16
\]

As the operation is done with 4 bits, adding 6 is equivalent to subtracting 10.

\( C_{OUT} = 1 \)

(iii) \( C_0 = 1 \): in this case \( z = (A + B + C_{in}) - 16 \) and we want \( s = (A + B + C_{in}) \mod 10 \). For this range of values we have \( (A + B + C_{in}) \mod 10 = (A + B + C_{in}) - 10 \), so we make:

\[
s = z + 6
\]

\( C_{OUT} = 1 \)

Therefore, to obtain the BCD adder, the following operations must be performed to the binary adder output \( z \):

\[
s = \begin{cases} 
  z & \text{if } z < 9 \text{ and } C_0 = 0 \\
  (z + 6) \mod 16 & \text{if } 10 \leq z \leq 15 \text{ or } C_0 = 1
\end{cases}
\]

\[
C_{OUT} = \begin{cases} 
  0 & \text{if } z < 9 \text{ and } C_0 = 0 \\
  1 & \text{if } z \geq 10 \text{ or } C_0 = 1
\end{cases}
\]

The condition \( z \geq 10 \) or \( C_0 = 1 \) is described by the switching expression:

\[
w = (z_1z_3 + z_2z_3 + C_0)
\]

The circuit is shown in Figure 10.2.

Exercise 10.5: Adder of decimal digits in Excess-3 code.