1. Wide fan-in XORs
   a. \( Z = A \oplus B \oplus C = A \oplus (B \oplus C) \)

   \[ t_{PD} = 2t_{PD,XOR2} \]

   ![Diagram of XOR gate with inputs A, B, and C, output Z.]

   \( t_{PD} = 2t_{PD,XOR2} \)

   b. \( Z = A \oplus B \oplus C \oplus D \oplus E = A \oplus B \oplus (C \oplus D \oplus E) \)

   \[ t_{PD} = 2t_{PD,XOR3} = 4t_{PD,XOR2} \]

   (This is a conservative estimate, however -- the delays from the different inputs to XOR3 are in fact asymmetrical, and so one could lay out the circuit to make the critical path shorter, \( t_{PD} = 3t_{PD,XOR2} \).)

   ![Diagram of XOR gate with inputs A, B, C, D, and E, output Z.]

   \( t_{PD} = 2t_{PD,XOR3} = 4t_{PD,XOR2} \)

   c. \( Z = A \oplus B \oplus C \oplus D \oplus E = ((A \oplus B) \oplus (C \oplus D)) \oplus E \)

   \[ t_{PD} = 3t_{PD,XOR2} \]

   ![Diagram of XOR gate with inputs A, B, C, D, and E, output Z.]

   \( t_{PD} = 3t_{PD,XOR2} \)

2. \( A \cdot \neg B \) is universal. We can construct a NOT as follows:

   ![Diagram of NOT gate with input A and output Z.]

   \( \neg A \)
We can use this NOT form to make AND and OR (the latter using DeMorgan's law):

\[ AND: \quad \begin{array}{cc}
A & \overline{A} \\
\downarrow & \downarrow \\
\rightarrow & \rightarrow \\
z & \overline{z}
\end{array}
\]

\[ OR: \quad \begin{array}{cc}
A & \overline{A} \\
\downarrow & \downarrow \\
\rightarrow & \rightarrow \\
z & \overline{z}
\end{array}
\]

Alternatively, we could just make NOR, which alone is sufficient to show universality, since NOR is universal:

\[ NOR: \quad \begin{array}{cc}
A & \overline{A} \\
\downarrow & \downarrow \\
\rightarrow & \rightarrow \\
z & \overline{z}
\end{array}
\]

One can see that this is a NOR by inspection with DeMorgan's law – push the two NOTs on the input of the last gate through to its output, which turns the AND into an OR.

3. A 2-input Elfen logic function maps from 5x5 possible input values to 5 output values. Each different truth table represents a different unique function, and so each row multiplies the number of combinations by 5. So, the total number of functions is the number of distinct truth tables, $5^{25}$.

4. Input | Output
--- | ---
0000 | 0
0001 | 0
0010 | 0
0011 | 1
0100 | 0
0101 | 1
0110 | 1
0111 | 0
1000 | 0
1001 | 1
1010 | 1
1011 | 0
1100 | 1
1101 | 0
1110 | 0
1111 | 0
(An aside: The SOP implementation above is rather hairy. Unfortunately, it cannot be simplified! This is characteristic of functions with "parity"-like behavior, including most arithmetic. It is possible, however, to build simpler versions using some combination of XORs and multi-level logic. For example, the above circuit can implemented as

\[ \text{abc} + \text{aBc} + \text{aBC} + \text{Abc} + \text{AbC} + \text{ABc} + \text{ABC} \]

(Simplified, this would be \( A + B + c \).)

5.

a. We can compute the truth table either by brute force substitution or by algebraic simplification to sum-of-products form.

<table>
<thead>
<tr>
<th>ABC</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
</tr>
</tbody>
</table>

b. \( \text{abc} + \text{aBc} + \text{aBC} + \text{Abc} + \text{AbC} + \text{ABc} + \text{ABC} \)
c. The deepest path is B->Z, passing through an INV, an ND2, and an ND3. The time sums up to $1.3 + 2.1 + 2.7 = 6.1\text{ns}$

d. We can construct a MUX8-based implementation as a literal representation of the truth table:

![MUX8 diagram]

6.

a. The longest path through the circuit is the one that traverses all three levels of NORs (two NORs and a NOT is shorter). This gives $t_{PD} = 3 \times 2\text{ns} = 6\text{ns}$

b. We can simplify this circuit either algebraically or by constructing a truth table and implementing that as simply as possible. Because this is only a two-input function, the truth table method should be easiest. (There are straightforward methods, not covered in this class, for implementing a truth table as two levels of logic in the simplest way possible. Here, though, it's simple enough to solve by inspection.)

<table>
<thead>
<tr>
<th>AB</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

It turns out this is just NOT(B). Implemented that way, it has $t_{PD} = 1\text{ns}$. 