**Decoder**

Function: sets exactly one of n outputs to 1, based on unsigned binary value  
Input: ceil (lg n)  
Output: n bits (exactly one is 1, rest are 0)

Example: 3-8 decoder  
Inputs: 3 bits representing UB number  
Output: 1 bit corresponding to the value of the UB number is set to 1

Truth table:
<table>
<thead>
<tr>
<th>$x_2$</th>
<th>$x_1$</th>
<th>$x_0$</th>
<th>$z_7$</th>
<th>$z_6$</th>
<th>$z_5$</th>
<th>$z_4$</th>
<th>$z_3$</th>
<th>$z_2$</th>
<th>$z_1$</th>
<th>$z_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Boolean expressions for each output:

- $z_0 = \overline{x_2} \overline{x_1} x_0$
- $z_4 = x_2 \overline{x_1} \overline{x_0}$
- $z_1 = \overline{x_2} x_1 x_0$
- $z_5 = x_2 \overline{x_1} x_0$
- $z_2 = \overline{x_2} x_1 \overline{x_0}$
- $z_6 = x_2 x_1 \overline{x_0}$
- $z_3 = \overline{x_2} x_1 x_0$
- $z_7 = x_2 x_1 x_0$
Decoder

Decoder vs. DEMUX

3-8 decoder: 3 data inputs, 8 outputs
1-8 DEMUX: 1 data input, 3 control inputs, 8 outputs

Add enable control bit to decoder:
  e = 0: all outputs are 0
  e = 1: behaves like regular decoder

Data inputs of a decoder correspond to the control bits of a DEMUX
Enable input of a decoder corresponds to the data bit of a DEMUX
The two circuits are identical.
Encoder

Encoder is reverse of decoder

8-3 encoder

8 inputs, exactly one has value 1
3 output bits, representing which input was equal to 1 (binary representation of input)
Example: input: $x_5 = 1$ output: $z_2z_1z_0 = 101$

Simplified truth table:

<table>
<thead>
<tr>
<th>Input $== 1$</th>
<th>$z_2$</th>
<th>$z_1$</th>
<th>$z_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$x_4$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_5$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Minterms are rather large

(from full truth table):

$z_2 = x_0x_1x_2x_3x_4x_5x_6x_7 +$
$\quad x_0x_1x_2x_3x_4x_5x_6x_7 +$
$\quad x_0x_1x_2x_3x_4x_5x_6x_7 +$
$\quad x_0x_1x_2x_3x_4x_5x_6x_7 +$
$\quad x_0x_1x_2x_3x_4x_5x_6x_7$

However, we can take advantage of the fact that exactly one input is 1:

$z_2 = x_4 + x_5 + x_6 + x_7$

Example: $x_5 = 1$

$z_1 = x_2 + x_3 + x_6 + x_7$

$z_0 = x_1 + x_3 + x_5 + x_7$

Example: $z_2z_1z_0 = 101$ (UB for 5)
Priority Encoder

We can't always assume that only one input will be 1. **Priority encoder** assumes that at least one input will be 1.

Which input to encode? Use priority scheme
- Larger subscripts could have higher priority
- Smaller subscripts could have higher priority

Assume larger subscripts have priority

Boolean expressions no longer necessarily valid

Suppose $x_4$ and $x_3$ are both equal to 1
- Then $z_2z_1z_0 = 111$, but the result should be $100$, since 4 has higher priority

What does it mean that 4 has the highest priority?
- All of the higher inputs must be 0, and the lower inputs don't matter:
  \[
  \neg x_7 x_6 x_5 x_4 \\
  \neg x_7 x_6 x_5 x_4 x_3 + \neg x_7 x_6 x_5 x_4 x_3 x_2 + x_7 x_6 x_5 x_4 x_3 x_2 \\
  \neg x_7 x_6 x_5 x_4 x_3 x_2 x_1 
  \]

Negate all literals with higher priority, and leave out lower ones

Replace each term in original expressions

- $z_2 = x_7 + \neg x_7 x_6 + \neg x_7 x_6 x_5 + \neg x_7 x_6 x_5 x_4$
- $z_1 = x_7 + \neg x_7 x_6 + \neg x_7 x_6 x_5 x_4 x_3 + \neg x_7 x_6 x_5 x_4 x_3 x_2$
- $z_0 = x_7 + \neg x_7 x_6 x_5 + \neg x_7 x_6 x_5 x_4 x_3 + \neg x_7 x_6 x_5 x_4 x_3 x_2 x_1$

This can be further simplified.

If $x_7$ is the highest priority 1, then it doesn't matter if the other terms are 0 or not.

- $z_2 = x_7 + x_6 + \neg x_6 x_5 + \neg x_6 x_5 x_4$
Similarly, if $x_6$ is the highest priority 1, then $\neg x_6$ is not necessary in the other 2 terms.

$$z_2 = x_7 + x_6 + x_5 + \neg x_5 x_4$$

We can also eliminate $\neg x_5$ in the last term.

$$z_2 = x_7 + x_6 + x_5 + x_4$$

(Notice that this expression gives back the original form.)

In the expression for $z_1$, however, we need to keep $\neg x_5$ and $\neg x_4$:

$$z_1 = x_7 + x_6 + \neg x_5 x_4 x_3 + \neg x_5 \neg x_4 x_2$$

Likewise, for $z_0$, we need to keep $\neg x_6$, $\neg x_4$, and $\neg x_2$:

$$z_0 = x_7 + \neg x_6 x_5 + \neg x_6 \neg x_4 x_3 + \neg x_6 \neg x_4 \neg x_2 x_1$$

In general, we need to keep the negation of any literal which doesn't appear as a higher-priority value.

What if all inputs are 0?

We can encode the output as 000, and $x_6$ will have the highest priority by default.