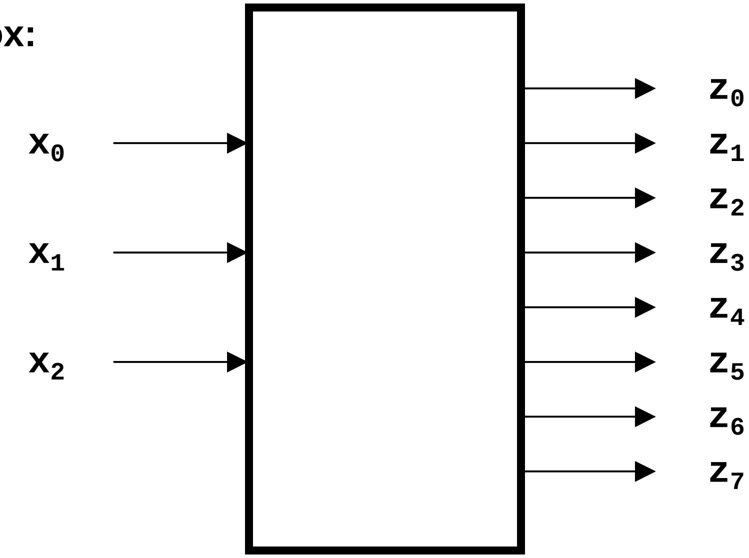
# Decoder

**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**

**Black box:**




**Truth table:**

| $x_2$ | $x_1$ | $x_0$ | $z_7$ | $z_6$ | $z_5$ | $z_4$ | $z_3$ | $z_2$ | $z_1$ | $z_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Boolean expressions for each output:**

$z_0 = \backslash x_2 \backslash x_1 \backslash x_0$

$z_1 = \backslash x_2 \backslash x_1 x_0$

$z_2 = \backslash x_2 x_1 \backslash x_0$

$z_3 = \backslash x_2 x_1 x_0$

$z_4 = x_2 \backslash x_1 \backslash x_0$

$z_5 = x_2 \backslash x_1 x_0$

$z_6 = x_2 x_1 \backslash x_0$

$z_7 = x_2 x_1 x_0$

## 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:** $z_2 z_1 z_0 = 101$

**Simplified truth table:**

| Input == 1 | $z_2$ | $z_1$ | $z_0$ |
|---|---|---|---|
| $x_0$ | 0 | 0 | 0 |
| $x_1$ | 0 | 0 | 1 |
| $x_2$ | 0 | 1 | 0 |
| $x_3$ | 0 | 1 | 1 |
| $x_4$ | 1 | 0 | 0 |
| $x_5$ | 1 | 0 | 1 |
| $x_6$ | 1 | 1 | 0 |
| $x_7$ | 1 | 1 | 1 |

**Minterms are rather large (from full truth table):**

$$z_2 = \backslash x_0\backslash x_1\backslash x_2\backslash x_3 x_4\backslash x_5\backslash x_6\backslash x_7 + \backslash x_0\backslash x_1\backslash x_2\backslash x_3\backslash x_4 x_5\backslash x_6\backslash x_7 + \backslash x_0\backslash x_1\backslash x_2\backslash x_3\backslash x_4\backslash x_5 x_6\backslash x_7 + \backslash x_0\backslash x_1\backslash x_2\backslash x_3\backslash x_4\backslash x_5\backslash x_6 x_7$$

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

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

$$z_1 = x_2 + x_3 + x_6 + x_7$$

$$z_0 = x_1 + x_3 + x_5 + x_7$$

**Example:** $x_5 = 1$

$z_2 z_1 z_0 = 101$

**(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:
  - $\backslash x_7\backslash x_6\backslash x_5x_4$
  - Negate all literals with higher priority, and leave out lower ones

Replace each term in original expressions

$$z_2 = x_7 + \backslash x_7x_6 + \backslash x_7\backslash x_6x_5 + \backslash x_7\backslash x_6\backslash x_5x_4$$

$$z_1 = x_7 + \backslash x_7x_6 + \backslash x_7\backslash x_6\backslash x_5\backslash x_4x_3 + \backslash x_7\backslash x_6\backslash x_5\backslash x_4\backslash x_3x_2$$

$$z_0 = x_7 + \backslash x_7\backslash x_6x_5 + \backslash x_7\backslash x_6\backslash x_5\backslash x_4x_3 + \backslash x_7\backslash x_6\backslash x_5\backslash x_4\backslash x_3\backslash x_2x_1$$

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.

$$z_2 = x_7 + x_6 + \backslash x_6x_5 + \backslash x_6\backslash x_5x_4$$

Similarly, if x6 is the highest priority 1, then $\backslash x_6$ is not necessary in the other 2 terms.

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

We can also eliminate $\backslash x_5$ 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 $\backslash x_5$ and $\backslash x_4$:

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

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

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

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 $x_0$ will have the highest priority by default.