16 ANDs + 3 inverters

If we only have inverters and standard 2-input ANDs, how many gates are needed to build a 3:8 decoder?
Decoders in General Logic Implementation

Implementation \( F = WXX + YZ \) w/4x16 Decoder
Function: the output is the binary representation of the ID of the input line that is asserted.

Output: \( N \) output lines

Given time: \( 2^N \) or less lines -- only 1 is asserted at any given time.

Performs the inverse operation of decoders.

Encoders
Problems:

\[ A_0 = D_3 + D_1 \]
\[ A_1 = D_3 + D_2 \]

<table>
<thead>
<tr>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
<th>A0</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td>X</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Input active**

Also add an output to identify when at least 1

**One scheme: Highest ID active wins**

Input lines active at a time.

Use priorities to resolve the problem of 2 or more

---

Priority Encoder
Priority Encoder Implementation (cont.)
module basicEncoder4_2 (out, in);
  output logic [1:0] out;
  input logic [3:0] in;

  always_comb begin
    assert (in == 4'b0001 || in == 4'b0010 || in == 4'b0100 || in == 4'b1000);
  end

  always_comb begin
    case (in)
      4'b0001: out = 2'b00;
      4'b0010: out = 2'b01;
      4'b0100: out = 2'b10;
      4'b1000: out = 2'b11;
      default: out = 2'bxx;
    endcase
  end

endmodule
val $d = Cini| = 4.9,00000$

\[
\begin{align*}
\text{out}_t &= 2.5 \times x \\ 
\text{out}_t &= 2.600 \\
\text{out}_t &= 2.010 \\
\text{out}_t &= 2.911 \\
\text{always-comb begin}
\end{align*}
\]

always-comb begin

val $t$ = (\text{input logic}) 
val $i$ = ($i_{in}$, $i_{out}$) 
output $logic$ [3:0] $i_{in}$; 
output $logic$ [1:0] $i_{out}$; 
module priority-encoder4,2 (out, in, valid);}

Priority Encoders in Verilog