Review Problem 41

- For the 6-bit 2’s Complement number (111010)
  - How would it be represented in 10-bit 2’s Complement?

\[ \begin{array}{c}
\begin{array}{c}
111111010 \\
-6
\end{array}
\end{array} \]

- What’s the smallest number of bits required to represent that number in 2’s Complement?

\[ \begin{align*}
1010 &= -(-1010) \\
 &= -(0110 + 1) \\
 &= -6
\end{align*} \]
Basic Circuit Elements

- **Readings:** 4-4.1.1, 4.2, 4.3-4.3.2
- **Standard TTL Small-Scale Integration:**
  1 chip = 2-8 gates
  - Requires numerous chips to build interesting circuits
- **Alternative:** Complex chips for standard functions
  - Single chip that performs very complex computations
- **Multiplexer/Decoder/Encoder:** Standard routing elements for interconnections
- **FPGAs:** Programmable for arbitrary functions
Design Example: Basic Telephone System

- Put together a simple telephone system
Basic Telephone System (cont.)

- Multiple subscribers, one operator.
- Operator controls all connections
Standard Circuit Elements

- Develop implementations of important “Building Blocks”
  - Used in Networks, Computers, Stereos, etc.
- Multiplexer: Combine N sources onto 1 wire
- Encoder: Determine which input is active
- Decoder: Convert binary to one-of-N wires

"code" is a binary #.
Decoders

- Used to select one of $2^N$ outputs based on N input bits
- Input: N bits; output: $2^N$ outputs -- only one is true
- A decoder that has $n$ inputs and $m$ outputs is referred to as an $n \times m$, $N:M$, or $n$-to-$m$ decoder
- Example: 3-to-8 decoder

$N:2^N$
Decoder Implementation

<table>
<thead>
<tr>
<th>S1</th>
<th>S0</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</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>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ D_3 = \overline{S_1} \cdot S_0 \]
\[ D_2 = \overline{S_1} \cdot \overline{S_0} \]
\[ D_1 = S_1 \cdot \overline{S_0} \]
\[ D_0 = S_1 \cdot S_0 \]
## Enabled Decoder Implementation

### Active High enable

<table>
<thead>
<tr>
<th>En</th>
<th>S1</th>
<th>S0</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</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>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</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>
</tr>
<tr>
<td>0</td>
<td>1</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
module enDecoder2_4 (out, in, enable);
    output logic [3:0] out;
    input logic [1:0] in;
    input logic enable;

always_comb begin
    if (enable)
        case (in)
            2'b00: out = 4'b00001;
            2'b01: out = 4'b0010;
            2'b10: out = 4'b0100;
            2'b11: out = 4'b1000;
        else
            out = 4'b0000;
end

endmodule
Decoder Expansion

- Construct a 4:16 decoder using 2:4 decoders