Encoders versus Decoders

- An encoder performs the inverse function as a decoder
- The simplest encoder to build is a $2^n$-to-$n$ (binary encoder)
Example: A decimal-to-BCD encoder

- A decimal-to-BCD encoder
  - Inputs: 10 bits corresponding to decimal digits 0 through 9, \((D_0, \ldots, D_9)\)
  - Outputs: 4 bits with BCD codes
  - Function: If input bit \(D_i\) is a “1”, then the output \((A_3, A_2, A_1, A_0)\) is the BCD code for \(i\)

![Diagram of decimal-to-BCD encoder]

Example: “7”, “3”, “8” \(\rightarrow\) “0111”, “0011”, “1000”

Truth table of the decimal-to-BCD encoder

- From the truth table, encoder outputs:
  - \(A_3 = D_8 + D_9\)
  - \(A_2 = D_4 + D_5 + D_6 + D_7\)
  - \(A_1 = D_2 + D_3 + D_6 + D_7\)
  - \(A_0 = D_1 + D_3 + D_5 + D_7 + D_9\)

- We made use of the fact that only one input can be “1” at one time
- Note that if none button is pushed, output is also “0000”
- What if two buttons are pushed simultaneously? —E.g. \(D_1\) and \(D_2\) together: \(A_0 = A_1 = 1\) and \(A_2 = A_3 = 0\) (0011) which is the same as if \(D_3\) were pushed!

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D9 D8 D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>A3 A2 A1 A0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 1</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 1 0</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1 0 0</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0 0 0</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 1 0 0 0 0</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>0 0 0 1 0 0 0 0 0 0</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>0 0 1 0 0 0 0 0 0 0</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>0 1 0 0 0 0 0 0 0 0</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>1 0 0 0 0 0 0 0 0 0</td>
<td>1 0 0 1</td>
</tr>
</tbody>
</table>
Binary Encoders

- General structure:

- 8-to-3 encoder:

```
\[
Y_0 = I_1 + I_3 + I_5 + I_7 \\
Y_1 = I_2 + I_3 + I_6 + I_7 \\
Y_3 = I_4 + I_5 + I_6 + I_7
\]
```

Priority Encoders

- If more than one input value is “1”, then the encoder just designed does not work properly.
- An encoder that can accept all possible combinations of input values and produce a meaningful result is a priority encoder.
- Among the “1”s that appear, it selects the most significant input position (or the least significant input position) containing a “1” and produces the corresponding binary code for that position.
- A system with \(2^n\) requestors and a “request encoder” that indicates which request signal is asserted at any time:
Exercise: Design a 4-input priority encoder with active low inputs

- Highest priority is given to most significant input “1” present (I3 … I0)
- Code outputs: A1, A0 and IDLE (indicates no input present)

Intermediate variables:
- H3 = I3'
- H2 = I2·I3
- H1 = I1'·I2·I3
- H0 = I0'·I1·I2·I3

Encoder outputs:
- A1 = H2 + H3
- A0 = H1 + H3
- IDLE = I3·I2·I1·I0

Truth table:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3 I2 I1 I0</td>
<td>A1 A0 IDLE</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>x x 1</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 1 0 x</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 0 x x</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 x x x</td>
<td>1 1 0</td>
</tr>
</tbody>
</table>

Exercise: Design a 4-input priority encoder with active low inputs

- Highest priority is given to most significant input “1” present (I3 … I0)
- Code outputs: A1, A0 and IDLE (indicates no input present)
- We could use a Karnaugh map to get equations, but can be read directly from the truth table and manually optimized if careful:

A1 = I3·I2' + I3' = I2' + I3'
A0 = I3·I2·I1' + I3' = I2·I1' + I3'
IDLE = I3·I2·I1·I0

Truth table:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3 I2 I1 I0</td>
<td>A1 A0 IDLE</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>x x 1</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 1 0 x</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 0 x x</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 x x x</td>
<td>1 1 0</td>
</tr>
</tbody>
</table>
8-input Priority Encoder

- Logic symbol for a generic 8-input priority encoder

A Generic 8-input Priority Encoder

- Truth table
**Priority-Encoder Logic Equations**

**Define intermediate variables**, s.t. \( H_i \) is “1” if and only if \( I_i \) is the highest-priority input:

\[
\begin{align*}
H_7 &= I_7 \\
H_6 &= I_6 \cdot I_7' \\
H_5 &= I_5 \cdot I_6' \cdot I_7' \\
&\vdots \\
H_0 &= I_0 \cdot I_1' \cdot I_2' \cdot I_3' \cdot I_4' \cdot I_5' \cdot I_6' \cdot I_7'
\end{align*}
\]

**Encoder outputs:**

\[
\begin{align*}
A_2 &= H_4 + H_5 + H_6 + H_7 \\
A_1 &= H_2 + H_3 + H_6 + H_7 \\
A_0 &= H_1 + H_3 + H_5 + H_7
\end{align*}
\]

The IDLE output is “1” if no inputs are “1”:

\[
\text{IDLE} = (I_0 + I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7)' = I_0' \cdot I_1' \cdot I_2' \cdot I_3' \cdot I_4' \cdot I_5' \cdot I_6' \cdot I_7'
\]

---

**74x148 8-input Priority Encoder**

- Active-low I/O
- \( EI = \) Enable Input
- \( GS = \) Group Select ("Got Something", as opposed to IDLE="Got Nothing")
  - One or more of the request inputs are asserted and \( EI \) is asserted
- \( EO = \) Enable Output
  - No request input is asserted and \( EI \) is asserted (corresponds to IDLE)
# Truth Table for a 74x148 Encoder

![Truth Table](image)

## 74x148 Logic Circuit

- **74x148**:
  - Enable Input (EI) must be asserted for any output to be asserted
  - Group Select (GS) is asserted if ≥1 inputs are asserted and EI is asserted
  - Enable Output (EO) is asserted if no input is asserted and EI is asserted
Example 74x148 Outputs

- 74x148 the A0_L output:
  \[ A0_L = 0 \text{ for } I7_L, I5_L, I3_L, I1_L \]
  \[ A0_L = (E \cdot I7 + E \cdot I6_L \cdot I5 + E \cdot I6_L \cdot I4_L \cdot I3 + \right. \]
  \[ + E \cdot I6_L \cdot I4_L \cdot I2_L \cdot I1)’ \]

- 74x148 the A1_L output:
  \[ A1_L = 0 \text{ for } I7_L, I6_L, I3_L, I2_L \]
  \[ A1_L = (E \cdot I7 + E \cdot I6 + E \cdot I5_L \cdot I4_L \cdot I3 + \right. \]
  \[ + E \cdot I5_L \cdot I4_L \cdot I2)’ \]

Cascading Priority Encoders

- 32-input priority encoder using four cascaded 74x148s
Cascading Priority Encoders

- EO_L is used for cascading to Enable Input of a lower-priority 74x148
- If none of the requests on the highest priority encoder is asserted (i.e., it’s IDLE), the next decoder in the cascade is enabled, etc.
- RA4-RA0 encode the highest-priority requestor

Some outputs of the four 74x148 cascade

- RA3 is “1” when 8–15 and 24–31 lines are requesting
  \[ RA3 = G1 \cdot GS + G3 \cdot GS \]

- RA1 is “1” when the second digit of the request is “1”
  \[ RA1 = G0 \cdot A1 + G1 \cdot A1 + G2 \cdot A1 + G3 \cdot A1 \]

- RCS is “1” when there is a request
  \[ RGS = G0 \cdot GS + G1 \cdot GS + G2 \cdot GS + G3 \cdot GS \]