ENGIN 112

Intro to Electrical and Computer Engineering

Lecture 17

Encoders and Decoders
Overview

° Binary decoders
  • Converts an n-bit code to a single active output
  • Can be developed using AND/OR gates
  • Can be used to implement logic circuits.

° Binary encoders
  • Converts one of $2^n$ inputs to an n-bit output
  • Useful for compressing data
  • Can be developed using AND/OR gates

° Both encoders and decoders are extensively used in digital systems
Binary Decoder

- Black box with \( n \) input lines and \( 2^n \) output lines
- Only one output is a 1 for any given input
2-to-4 Binary Decoder

Truth Table:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>F₀</th>
<th>F₁</th>
<th>F₂</th>
<th>F₃</th>
</tr>
</thead>
<tbody>
<tr>
<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>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

° From truth table, circuit for 2x4 decoder is:

° Note: Each output is a 2-variable minterm (X'Y', X'Y, XY' or XY)
3-to-8 Binary Decoder

Truth Table:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>F_0</th>
<th>F_1</th>
<th>F_2</th>
<th>F_3</th>
<th>F_4</th>
<th>F_5</th>
<th>F_6</th>
<th>F_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
<td>1</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>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</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>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>1</td>
<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>
</tr>
<tr>
<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>
<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>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

F_0 = x'y'z'
F_1 = x'y'z
F_2 = x'yz'
F_3 = x'yz
F_4 = xy'z'
F_5 = xy'z
F_6 = xyz'
F_7 = xyz
Implementing Functions Using Decoders

° Any n-variable logic function can be implemented using a single n-to-$2^n$ decoder to generate the minterms
  • OR gate forms the sum.
  • The output lines of the decoder corresponding to the minterms of the function are used as inputs to the or gate.

° Any combinational circuit with $n$ inputs and $m$ outputs can be implemented with an n-to-$2^n$ decoder with $m$ OR gates.

° Suitable when a circuit has many outputs, and each output function is expressed with few minterms.
Example: Full adder

\[ S(x, y, z) = \Sigma (1, 2, 4, 7) \]
\[ C(x, y, z) = \Sigma (3, 5, 6, 7) \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</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>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

![3-to-8 Decoder Diagram]
Standard MSI Binary Decoders Example

74138 (3-to-8 decoder)

(a) Logic circuit.
(b) Package pin configuration.
(c) Function table.
Building a Binary Decoder with NAND Gates

- Start with a 2-bit decoder
  - Add an enable signal (E)

Note: use of NANDs

only one 0 active!

if E = 0

(a) Logic diagram

(b) Truth table

Fig. 4-19 2-to-4-Line Decoder with Enable Input
Use two 3 to 8 decoders to make 4 to 16 decoder

- Enable can also be active high
- In this example, only one decoder can be active at a time.
- \( x, y, z \) effectively select output line for \( w \)

![Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders](image)
Encoders

° If the a decoder's output code has fewer bits than the input code, the device is usually called an encoder.

  e.g. $2^n$-to-$n$

° The simplest encoder is a $2^n$-to-$n$ binary encoder
  • One of $2^n$ inputs = 1
  • Output is an $n$-bit binary number
8-to-3 Binary Encoder

At any one time, only one input line has a value of 1.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7</td>
<td>y_2 y_1 y_0</td>
</tr>
<tr>
<td>1 0 0 0 0 0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 0 0 0 0 0 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 0 1 0 0 0 0 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0 0 0 1 0 0 0 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0 0 0 0 1 0 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 1 0 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

\[ y_2 = I_4 + I_5 + I_6 + I_7 \]

\[ y_1 = I_2 + I_3 + I_6 + I_7 \]

\[ y_0 = I_1 + I_3 + I_5 + I_7 \]
8-to-3 Priority Encoder

- What if more than one input line has a value of 1?
- Ignore “lower priority” inputs.
- **Idle** indicates that no input is a 1.
- Note that polarity of **Idle** is opposite from Table 4-8 in Mano

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₀ I₁ I₂ I₃ I₄ I₅ I₆ I₇</td>
<td>y₂ y₁ y₀ Idle</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>x x x 1</td>
</tr>
<tr>
<td>1 0 0 0 0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>X 1 0 0 0 0 0 0</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>X X 1 0 0 0 0 0</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>X X X 1 0 0 0 0</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>X X X X 1 0 0 0</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>X X X X X 1 0 0</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>X X X X X X X 1 0</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>X X X X X X X 1</td>
<td>1 1 1 0</td>
</tr>
</tbody>
</table>
Priority Encoder (8 to 3 encoder)

- Assign priorities to the inputs
- When more than one input are asserted, the output generates the code of the input with the highest priority

**Priority Encoder:**

- $H_7 = I_7$ (Highest Priority)
- $H_6 = I_6 \cdot I_7'$
- $H_5 = I_5 \cdot I_6' \cdot I_7'$
- $H_4 = I_4 \cdot I_5' \cdot I_6' \cdot I_7'$
- $H_3 = I_3 \cdot I_4' \cdot I_5' \cdot I_6' \cdot I_7'$
- $H_2 = I_2 \cdot I_3' \cdot I_4' \cdot I_5' \cdot I_6' \cdot I_7'$
- $H_1 = I_1 \cdot I_2' \cdot I_3' \cdot I_4' \cdot I_5' \cdot I_6' \cdot I_7'$
- $H_0 = I_0 \cdot I_1' \cdot I_2' \cdot I_3' \cdot I_4' \cdot I_5' \cdot I_6' \cdot I_7'$

**Encoder**

- $Y_0 = I_1 + I_3 + I_5 + I_7$
- $Y_1 = I_2 + I_3 + I_6 + I_7$
- $Y_2 = I_4 + I_5 + I_6 + I_7$
Encoder Application (Monitoring Unit)

- Encoder identifies the requester and encodes the value
- Controller accepts digital inputs.
Summary

° Decoder allows for generation of a single binary output from an input binary code
  • For an $n$-input binary decoder there are $2^n$ outputs

° Decoders are widely used in storage devices (e.g. memories)
  • We will discuss these in a few weeks

° Encoders all for data compression

° Priority encoders rank inputs and encode the highest priority input

° Next time: storage elements!