Topics

- Multiplexers/Demultiplexers
- Decoders/Encoders
- Comparators
- Shifters

Logic Circuit Types

- **Combinational**
  - Output is a function of inputs ONLY
  - e.g. briefcase lock

- **Sequential**
  - Output is a function of inputs and previous state (memory)
  - e.g. vending machine

Multiplexers

- **Signals**
  - $2^n$ data inputs
  - n control (select) inputs
  - 1 data output

- **Binary code on control lines determines which input is connected (gated/routed) to output**

- **Example: 2:1 Mux**

  ![Graphical symbol](image)
  ![Truth table](image)
  ![Sum-of-products circuit](image)

  $f = s'w_0 + sw_1$

Adapted from Aleksandar Milenkovic's CPE/EE 422/522 lecture notes. Copyright © 2003.
4:1 Multiplexer

\[ f = S_1' S_0' W_0 + S_1' S_0 W_1 + S_1 S_0' W_2 + S_1 S_0 W_3 \]

Demultiplexer

- Routes an input signal to one of \(2^n\) outputs
- Implementation is similar to decoder with enable

\[
\begin{array}{c|ccccc}
S_1 & S_0 & Y_0 & Y_1 & Y_2 & Y_3 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Decoder

- Takes an n-bit number as an input and selects exactly 1 of \(2^n\) outputs
- Potential uses:
  - Choosing a memory bank with the decoder as an "enable"
  - Accessing a register location in the register file with the decoder as the "enable"
- Another use for decoders:
  - the output of the decoders are merely the minterms of the inputs
  - you can make any function by merely ORing the appropriate minterms
### 2:4 Decoder with Enable

#### (a) Truth table

<table>
<thead>
<tr>
<th>$En$</th>
<th>$w_3$</th>
<th>$w_2$</th>
<th>$y_0$</th>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$y_3$</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>
</tr>
<tr>
<td>0</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>1</td>
<td>0</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>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>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>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>
</tbody>
</table>

#### (b) Graphic symbol

#### (c) Logic circuit

### Encoders

- **Opposite of decoders**
  - Encode given information into a more compact form
- **Binary encoders**
  - $2^n$ inputs into $n$-bit code
  - Exactly one of the input signals should have a value of 1, and outputs present the binary number that identifies which input is equal to 1
- **Use:** reduce the number of bits (transmitting and storing information)

#### 3:8 Decoder

![3:8 Decoder](image)

Figure 3.13. A 3-to-8 decoder circuit.

### 4:2 Encoder

#### (a) Truth table

<table>
<thead>
<tr>
<th>$w_3$</th>
<th>$w_2$</th>
<th>$w_1$</th>
<th>$w_0$</th>
<th>$y_1$</th>
<th>$y_0$</th>
</tr>
</thead>
<tbody>
<tr>
<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>1</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>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

#### (b) Circuit
Encoders: Priority Encoders

- Each input has a priority level associated with it
- The encoder outputs indicate the active input that has the highest priority

(a) Truth table for a 4-to-2 priority encoder

<table>
<thead>
<tr>
<th>W3</th>
<th>W2</th>
<th>W1</th>
<th>W0</th>
<th>Y1</th>
<th>Y0</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Adapted from Aleksandar Milenkovic’s CPE/EE 422/522 lecture notes. Copyright © 2003.

Comparator

- Determine if two input words are equal
- Example
  - Used to compare TAG field from cache accesses
- Uses the XOR/XNOR gate as the building block

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Truth table for XOR function

Summary

- Combinational logic blocks are used to provide useful functions with limited number of external connections
- Decoders and muxes can be used to implement truth tables
- Comparators can be a costly function because they are constructed of XOR gates

Shifters

- Left shift is the same as multiplying by powers of 2
- Right shift is the same as dividing by powers of 2
  - Odd numbers lose the last bit
- Arithmetic shift maintains the sign
- Logical shift fills empty bits with zeroes

Figure 5-16. A 1-bit left-right shifter.