Engr354: Digital Logic Circuits

Chapter 6: Combinational Blocks

Dr Curtis Nelson

Chapter 6 Overview

In this chapter you will learn about:
• Commonly used combinational sub-circuits;
• Multiplexers, which can be used for selection of signals and for implementation of general logic functions;
• Circuits for encoding, decoding, and code-conversion purposes.
Combinational Circuits

- Combinational circuits are those whose outputs are completely defined by the inputs, i.e. no memory is involved.
- Examples include
  - Basic gates like Nands, Nors and Inverters;
  - Multiplexers, de-multiplexers;
  - Decoders, encoders, code converters;
  - Arithmetic circuits;
  - Magnitude comparators;
  - Parity checkers;
  - Etc.

Multiplexers

- Devices that select one of many inputs to be routed to one output based on the binary value of select lines
  - Enable – used to enable or disable the complete function;
  - Select – used to select which one of the inputs gets routed to the output.

\[
\begin{array}{c}
\text{2}^n \text{ inputs} \\
\{i_0, \ldots, i_{n-1}\} \\
\text{Enable} \\
\text{n select lines} \\
\end{array} \quad \begin{array}{c}
\text{select} \\
\text{En} \\
\text{output} \\
y_0 \\
\end{array}
\]
2-to-1 Multiplexer

(a) Graphical symbol

(b) Truth table

(c) Sum-of-products circuit

4-to-1 Multiplexer

(a) Graphical symbol

(b) Truth table

(c) Circuit
Using 2-to-1 Mux’s to Build a 4-to-1 Mux

Practical Application

(a) A 2x2 crossbar switch

(b) Implementation using multiplexers
Multiplexer Data Sheet

- 74HC153 Dual 4-Line to 1-Line Data Selectors/Multiplexers

Synthesis of Logic Functions Using Mux’s

- Mux’s can be used to synthesize logic functions as follows
  - Create truth table;
  - Compress as necessary;
  - Implement.
- In general, an $N$ variable function can be implemented with one $N-1$ multiplexer and at most, one inverter.
Example 2-Input Function

(a) Implementation using a 4-to-1 multiplexer

(b) Modified truth table

(c) Circuit

Example 3-Input Majority Function

(a) Modified truth table

(b) Circuit
Example 3-Input Majority Function

\[
\begin{array}{c|c|c|c}
w_1 & w_2 & w_3 & f \\
\hline
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

(a) Truth table

\[
f = w_1 + w_2 w_3
\]

(b) Circuit

Example 3-Input XOR Function

\[
\begin{array}{c|c|c|c}
w_1 & w_2 & w_3 & f \\
\hline
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

(a) Truth table

\[
f = w_3 \oplus w_1 w_2
\]

(b) Circuit
De-Multiplexers

- A de-multiplexer is a circuit which places the value of a single data input onto one of a number of outputs.

![De-Multiplexer Circuit](image)

An \(n\)-to-\(2^n\) Decoder

- A decoder is a device that activates one output, whose outputs are usually active low, based on the binary value of the inputs;
- A decoder is a minterm generator;
- Enable – used to enable or disable the complete decoder function.

![Decoder Circuit](image)
A 2-to-4 Decoder

(a) Truth table

<table>
<thead>
<tr>
<th></th>
<th>w0</th>
<th>w1</th>
<th>y0</th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<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>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>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Graphic symbol

(c) Logic circuit

3-to-8 Decoder Using 2-to-4 Decoders
Decoder Tree

Decoder Data Sheet

- 74HC138 Single 3-Line to 8-Line Decoders/Demultiplexers
Combinational Design with Decoders

- Implement the function \( f(a,b,c) = \Sigma m(0,3,4,5) \) using a decoder and random logic.
- Implement the function \( f(a,b,c) = \Pi M(0,1,4,6) \) using a decoder and random logic.

Combinational Design with Decoders and Mux’s

- Design a digital circuit which has a 4-bit input, \( A(H) = A_3A_2A_1A_0 \) and a single output \( Z(H) \). \( Z(H) \) is high if \( A(H) \) is exactly divisible by 3 and \( A(H) \) is not equal to 12. Note that 0 is not divisible by 3 in this circuit. Use a 2 to 4 decoder, a 4 to 1 MUX, and minimum logic gates, if necessary. Note that all inputs and outputs of the MUX and decoder are asserted HIGH.
A $2^n$-to-$n$ Binary Encoder

- An encoder is a device that outputs a binary code representing which one of many inputs is active. The outputs are usually active low.
- Priority encoder – assigns priority to certain inputs
  - Used in embedded computer systems to service interrupts.

\[
\begin{align*}
\{ w_0, \ldots, w_{2^n-1} \} &\rightarrow \{ y_0, \ldots, y_{n-1} \} \\
2^n \quad \text{inputs} &\quad n \quad \text{outputs}
\end{align*}
\]

A 4-to-2 Binary 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>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) Circuit
Truth Table for a 4-to-2 Priority Encoder

- Create a circuit using the following truth table for a 4-to-2 priority encoder (z is an output that is asserted whenever any output is asserted).

<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>
<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>0</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>

BCD to 7-segment Code Converter

(a) Code converter
(b) 7-segment display

<table>
<thead>
<tr>
<th>(w_3)</th>
<th>(w_2)</th>
<th>(w_1)</th>
<th>(w_0)</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<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>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</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>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</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>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
A Four-bit Magnitude Comparator

Arithmetic Logic Units (74HC381 ALU)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Inputs $s_2$ $s_1$ $s_0$</th>
<th>Outputs $F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>B $\leftarrow$ A</td>
<td>0 0 1</td>
<td>B $\leftarrow$ A</td>
</tr>
<tr>
<td>A $\leftarrow$ B</td>
<td>0 1 0</td>
<td>A $\leftarrow$ B</td>
</tr>
<tr>
<td>ADD</td>
<td>0 1 1</td>
<td>A + B</td>
</tr>
<tr>
<td>XOR</td>
<td>1 0 0</td>
<td>A XOR B</td>
</tr>
<tr>
<td>OR</td>
<td>1 0 1</td>
<td>A OR B</td>
</tr>
<tr>
<td>AND</td>
<td>1 1 0</td>
<td>A AND B</td>
</tr>
<tr>
<td>Preset</td>
<td>1 1 1</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>
Summary

In this chapter you learned about:
• Commonly used combinational sub-circuits;
• Multiplexers, which can be used for selection of signals and for implementation of general logic functions;
• Circuits for encoding, decoding, and code-conversion purposes.