Chapter 2

Introduction to Logic Circuits
Figure 2.1. A binary switch.

(a) Two states of a switch

\[ x = 0 \quad \quad \quad x = 1 \]

(b) Symbol for a switch

Figure 2.1. A binary switch.
Figure 2.2. A light controlled by a switch.

(a) Simple connection to a battery

(b) Using a ground connection as the return path

Figure 2.2. A light controlled by a switch.
(a) The logical AND function (series connection)

(b) The logical OR function (parallel connection)

Figure 2.3. Two basic functions.
Figure 2.4. A series-parallel connection.
Figure 2.5. An inverting circuit.
Figure 2.6. A truth table for the AND and OR operations.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_1 \cdot x_2$</th>
<th>$x_1 + x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

AND  OR
Figure 2.7. Three-input AND and OR operations.
(a) AND gates

(b) OR gates

(c) NOT gate

Figure 2.8. The basic gates.
Figure 2.9. The function from Figure 2.4.
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure 2.10. An example of logic networks.
Figure 2.11. An example of a logic circuit.
Figure 2.12. Addition of binary numbers.

(a) Evaluation of $S = a + b$

(b) Truth table

(c) Logic network
Figure 2.13. Proof of DeMorgan’s theorem in 15a.
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure 2.14. The Venn diagram representation.
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure 2.15. Verification of the distributive property.
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure 2.16. Verification of $x\cdot y + \bar{x}\cdot z + y\cdot z = x\cdot y + \bar{x}\cdot z$. 
Figure 2.17. Proof of the distributive property \(12b\).
Figure 2.18. Proof of DeMorgan’s theorem 15a.
<table>
<thead>
<tr>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>$f(x_{11}, x_{12})$</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>
Figure 2.20. Two implementations of the function in Figure 2.19.
Figure 2.21. A bubble gumball factory.
<table>
<thead>
<tr>
<th>Row number</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>Minterm</th>
<th>Maxterm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$m_0 = \overline{x}_1 \overline{x}_2 \overline{x}_3$</td>
<td>$M_0 = x_1 + x_2 + x_3$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$m_1 = \overline{x}_1 \overline{x}_2 x_3$</td>
<td>$M_1 = x_1 + x_2 + \overline{x}_3$</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$m_2 = \overline{x}_1 x_2 \overline{x}_3$</td>
<td>$M_2 = x_1 + \overline{x}_2 + x_3$</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$m_3 = \overline{x}_1 x_2 x_3$</td>
<td>$M_3 = x_1 + \overline{x}_2 + \overline{x}_3$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$m_4 = x_1 \overline{x}_2 \overline{x}_3$</td>
<td>$M_4 = \overline{x}_1 + x_2 + x_3$</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$m_5 = x_1 \overline{x}_2 x_3$</td>
<td>$M_5 = \overline{x}_1 + x_2 + \overline{x}_3$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$m_6 = x_1 x_2 \overline{x}_3$</td>
<td>$M_6 = \overline{x}_1 + \overline{x}_2 + x_3$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$m_7 = x_1 x_2 x_3$</td>
<td>$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$</td>
</tr>
</tbody>
</table>

Figure 2.22 Three-variable minterms and maxterms.
<table>
<thead>
<tr>
<th>Row number</th>
<th>( z_1 )</th>
<th>( z_2 )</th>
<th>( z_3 )</th>
<th>( f(z_1, z_2, z_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 2.23.** A three-variable function.
Figure 2.24. Two realizations of a function in Figure 2.23.

(a) A minimal sum-of-products realization

(b) A minimal product-of-sums realization
Figure 2.25. NAND and NOR gates.
(a) $x_1 \overline{x_2} = \overline{x_1} + \overline{x_2}$

(b) $x_1 + x_2 = \overline{x_1} \overline{x_2}$

Figure 2.26. DeMorgan’s theorem in terms of logic gates.
Figure 2.27. Using NAND gates to implement a sum-of-products.
Figure 2.28. Using NOR gates to implement a product-of-sums.
Figure 2.29  NOR-gate realization of the function in Example 2.11.
Figure 2.30. NAND-gate realization of the function in Example 2.10.
<table>
<thead>
<tr>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>$x_{13}$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 2.32. Implementation of the function in Figure 2.31.

Please see “portrait orientation” PowerPoint file for Chapter 2
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure 2.33. Implementation of a multiplexer.
Figure 2.34. Display of numbers.

(a) Logic circuit and 7-segment display

(b) Truth table
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure 2.35. A typical CAD system.
Figure 2.36. The logic circuit for a multiplexer.
module example1 (x1, x2, s, f);
    input  x1, x2, s;
    output f;

    not (k, s);
    and (g, k, x1);
    and (h, s, x2);
    or (f, g, h);
endmodule
module example2 (x1, x2, x3, x4, f, g, h);
  input x1, x2, x3, x4;
  output f, g, h;

  and (z1, x1, x3);
  and (z2, x2, x4);
  or (g, z1, z2);
  or (z3, x1, ~x3);
  or (z4, ~x2, x4);
  and (h, z3, z4);
  or (f, g, h);

endmodule

Figure 2.38. Verilog code for a four-input circuit.
Figure 2.39. Logic circuit for the code in Figure 2.38.
module example3 (x1, x2, s, f);
    input x1, x2, s;
    output f;

    assign f = (~s & x1) | (s & x2);

endmodule

Figure 2.40. Using the continuous assignment to specify the circuit in Figure 2.36.
module example4 (x1, x2, x3, x4, f, g, h);
  input x1, x2, x3, x4;
  output f, g, h;

  assign g = (x1 & x3) | (x2 & x4);
  assign h = (x1 | ~x3) & (~x2 | x4);
  assign f = g | h;

endmodule

Figure 2.41. Using the continuous assignment to specify the circuit in Figure 2.39.
Figure 2.42. Behavioral specification of the circuit in Figure 2.36.

```verilog
// Behavioral specification
module example5 (x1, x2, s, f);
    input x1, x2, s;
    output f;
    reg f;

    always @(x1 or x2 or s)
        if (s == 0)
            f = x1;
        else
            f = x2;
endmodule
```
// Behavioral specification
module example5 (input x1, x2, s, output reg f);

always @(x1, x2, s)
    if (s == 0)
        f = x1;
    else
        f = x2;

endmodule

Figure 2.43. A more compact version of the code in Figure 2.42.
Figure 2.44. A logic circuit with two modules.
// An adder module
module adder (a, b, s1, s0);
  input a, b;
  output s1, s0;

  assign s1 = a & b;
  assign s0 = a ^ b;

endmodule

Figure 2.45. Verilog specification of the circuit in Figure 2.12.
// A module for driving a 7-segment display
module display (s1, s0, a, b, c, d, e, f, g);
    input s1, s0;
    output a, b, c, d, e, f, g;

    assign a = ~s0;
    assign b = 1;
    assign c = ~s1;
    assign d = ~s0;
    assign e = ~s0;
    assign f = ~s1 & ~s0;
    assign g = s1 & ~s0;
endmodule

Figure 2.46. Verilog specification of the circuit in Figure 2.34.
module adder_display (x, y, a, b, c, d, e, f, g);
    input x, y;
    output a, b, c, d, e, f, g;
    wire w1, w0;

    adder U1 (x, y, w1, w0);
    display U2 (w1, w0, a, b, c, d, e, f, g);

endmodule

Figure 2.47. Hierarchical Verilog code for the circuit in Figure 2.44.
The function \( f(x_1, x_2, x_3) = \sum m(0, 2, 4, 5, 6) \).
Figure 2.49. Location of two-variable minterms.
Figure 2.50. The function of Figure 2.19.
Figure 2.51. Location of three-variable minterms.

(a) Truth table

(b) Karnaugh map
Figure 2.52. Examples of three-variable Karnaugh maps.
Figure 2.53. A four-variable Karnaugh map.
Figure 2.54. Examples of four-variable Karnaugh maps.

\[ f_1 = \overline{x_2}x_3 + x_1\overline{x_3}x_4 \]
\[ f_2 = x_3 + x_1x_4 \]
\[ f_3 = \overline{x_2}\overline{x_4} + \overline{x_1}x_3 + x_2x_3x_4 \]
\[ f_4 = \overline{x_1}\overline{x_3} + x_1x_3 + \text{or} \quad \overline{x_2}\overline{x_3} \]
Figure 2.55. A five-variable Karnaugh map.
Figure 2.56. Three-variable function $f(x_1, x_2, x_3) = \Sigma m(0, 1, 2, 3, 7)$. 

<table>
<thead>
<tr>
<th></th>
<th>$x_1x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 2.57. Four-variable function \( f(x_1, \ldots, x_4) = \sum m(2, 3, 5, 6, 7, 10, 11, 13, 14) \).
Figure 2.58. The function $f(x_1, \ldots, x_4) = \sum m(0, 4, 8, 10, 11, 12, 13, 15)$. 

This describes a function $f$ defined on a 4-variable Boolean space, where $x_1, x_2, x_3, x_4$ are the variables, and the function is represented by a minterm sum of products. The diagram shows the truth table with corresponding minterms labeled, indicating which combinations of $x_1, x_2, x_3, x_4$ result in the function output being 1. The minterms are identified by their decimal equivalent: $m(0) = x_3\bar{x}_4, m(4) = \bar{x}_1x_2\bar{x}_3, m(8) = \bar{x}_1\bar{x}_2\bar{x}_4, m(10) = \bar{x}_1x_3x_4, m(11) = \bar{x}_1\bar{x}_2x_3, m(12) = \bar{x}_1x_2x_4$. The function is a product of these minterms, representing the Boolean function evaluated for all possible input combinations.
Figure 2.59. The function \( f(x_1, \ldots, x_4) = \Sigma m(0, 2, 4, 5, 10, 11, 13, 15) \).
Figure 2.60. POS minimization of \( f(x_1, x_2, x_3) = \Pi M(4, 5, 6). \)
Figure 2.61. POS minimization of \( f(x_1, \ldots, x_4) = \Pi M(0, 1, 4, 8, 9, 12, 15) \).
Figure 2.62. Two implementations of the function $f(x_1, \ldots, x_4) = \sum m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$.

Please see “portrait orientation” PowerPoint file for Chapter 2
Figure 2.63. Using don’t-care minterms when displaying BCD numbers.

Please see “portrait orientation” PowerPoint file for Chapter 2.
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure 2.64. An example of multiple-output synthesis.
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure 2.65. Another example of multiple-output synthesis.
Figure 2.66. The Venn diagrams for Example 2.23.
Figure 2.67. Karnaugh maps for Example 2.26.
Figure 2.68. Karnaugh maps for Example 2.27.

Please see “portrait orientation” PowerPoint file for Chapter 2
Figure 2.69. A K-map that represents the function in Example 2.28.
Figure 2.70. The logic circuit for Example 2.29.
module f_g (x, y, z, f, g);
    input x, y, z;
    output f, g;
    wire k;

    assign k = y ^ z;
    assign g = k ^ x;
    assign f = (~k & z) | (k & x);

endmodule

Figure 2.70. Verilog code for Example 2.29.
Figure 2.72. The circuit for Example 2.30.
module shared (a, b, c, d, m, s1, s0);
    input a, b, c, d, m;
    output s1, s0;
    wire w1, w2;
    mux2to1 U1 (a, c, m, w1);
    mux2to1 U2 (b, d, m, w2);
    adder U3 (w1, w2, s1, s0);
endmodule

module mux2to1 (x1, x2, s, f);
    input x1, x2, s;
    output f;
    assign f = (~s & x1) | (s & x2);
endmodule

module adder (a, b, s1, s0);
    input a, b;
    output s1, s0;
    assign s1 = a & b;
    assign s0 = a ^ b;
endmodule
Figure P2.1. Two attempts to draw a four-variable Venn diagram.
Figure P2.2. A four-variable Venn diagram.
Figure P2.3. A timing diagram representing a logic function.
Figure P2.4. A timing diagram representing a logic function.
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure P2.5. Circuit for problem 2.78.
Please see “portrait orientation” PowerPoint file for Chapter 2

Figure P2.6. Circuit for problem 2.79.