### Which Truth Tables Are the Same?

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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### The Idea of Min Term / Product Term

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>F</th>
<th>A</th>
<th>B</th>
<th>Min Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>0</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>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- Each row in a truth table represents a unique combination of variables.
- Each row can be expressed as a logic combination specifying when that row combination is equal to a 1.
- The term is called a **MIN TERM** or a **PRODUCT TERM**.
- Thus $F = A + B = X'Y' + XY$.

### Truth Table with Two Inputs

- Two inputs X and Y; Output is F.
- Logic Function: 
  \[ F = 1 \text{ if and only if } X = Y \]
- Truth Table:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>F</th>
<th>Min Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X'Y'</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X'Y</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>XY'</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>XY</td>
</tr>
</tbody>
</table>

- Logic Expression:
  \[ F = X'Y' + XY' + XY + X'Y \]

### Min / Product terms for more variables

<table>
<thead>
<tr>
<th>XYZ</th>
<th>Min Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>X'Y'Z'W'</td>
</tr>
<tr>
<td>001</td>
<td>X'Y'Z'W</td>
</tr>
<tr>
<td>010</td>
<td>X'Y'ZW'</td>
</tr>
<tr>
<td>011</td>
<td>X'Y'ZW</td>
</tr>
<tr>
<td>100</td>
<td>X'YZ'W'</td>
</tr>
<tr>
<td>101</td>
<td>X'YZW</td>
</tr>
<tr>
<td>110</td>
<td>XYZ'W'</td>
</tr>
<tr>
<td>111</td>
<td>XYZW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XYZW</th>
<th>Min Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>X'Y'Z'W'</td>
</tr>
<tr>
<td>0001</td>
<td>X'Y'Z'W</td>
</tr>
<tr>
<td>0010</td>
<td>X'Y'ZW'</td>
</tr>
<tr>
<td>0011</td>
<td>X'Y'ZW</td>
</tr>
<tr>
<td>0100</td>
<td>X'YZ'W'</td>
</tr>
<tr>
<td>0101</td>
<td>X'YZW</td>
</tr>
<tr>
<td>0110</td>
<td>XYZ'W'</td>
</tr>
<tr>
<td>0111</td>
<td>XYZW</td>
</tr>
<tr>
<td>1000</td>
<td>X'Y'Z'W'</td>
</tr>
<tr>
<td>1001</td>
<td>X'Y'ZW</td>
</tr>
<tr>
<td>1010</td>
<td>X'YZ'W</td>
</tr>
<tr>
<td>1011</td>
<td>X'YZW</td>
</tr>
<tr>
<td>1100</td>
<td>XYZ'W</td>
</tr>
<tr>
<td>1101</td>
<td>XYZW</td>
</tr>
<tr>
<td>1110</td>
<td>X'Y'Z'W</td>
</tr>
<tr>
<td>1111</td>
<td>X'Y'ZW</td>
</tr>
</tbody>
</table>

### Truth Table with Three Inputs

- Three inputs X, Y, and Z; Output is F.
- Logic Function:
  \[ F = 1 \text{ if and only if there is a 0 to the left of a 1 in the input} \]
- Truth Table:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>F</th>
<th>Min Term</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>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</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>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Logic Expression:
  \[ F = X'Y'Z' + X'YZ' + XYZ' + X'YZ' + X'Y'Z + XY'Z + X'YZ + XYZ \]

### Truth Table with Four Inputs

- Four inputs X, Y, Z, and W; Output is F.
- Logic Function:
  \[ F = 1 \text{ if and only if number of variables with value 1 is more than the number of variables with value 0} \]
- Truth Table:

<table>
<thead>
<tr>
<th>XYZW</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>0</td>
</tr>
<tr>
<td>0010</td>
<td>0</td>
</tr>
<tr>
<td>0011</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>0</td>
</tr>
<tr>
<td>0101</td>
<td>0</td>
</tr>
<tr>
<td>0110</td>
<td>1</td>
</tr>
<tr>
<td>0111</td>
<td>1</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1001</td>
<td>0</td>
</tr>
<tr>
<td>1010</td>
<td>0</td>
</tr>
<tr>
<td>1011</td>
<td>1</td>
</tr>
<tr>
<td>1100</td>
<td>0</td>
</tr>
<tr>
<td>1101</td>
<td>1</td>
</tr>
<tr>
<td>1110</td>
<td>1</td>
</tr>
<tr>
<td>1111</td>
<td>1</td>
</tr>
</tbody>
</table>
A product (min) term is a unique combination of variables:
- It has a value of 1 for only one input combination
- It is 0 for all the other combinations of variables

To write an expression, we need not write the entire truth table:
- We only need those combinations for which function output is 1

For example, for the function below:
\[ f = x'yz' + xy'z' + xyz \]

This is called the Canonical Sum-of-Product (SOP) Expression.

### Shorthand Notation for Canonical SOP

- We can also assign an integer to represent each input combination
- Thus the function produces a 1 for input combinations 2, 4, 7
- Therefore, the function can be written as \( f(x,y,z) = \sum m(2,4,7) \)

### Max / Sum Terms

- A max (sum) term is also a unique combination of variables
- However, it is opposite of a min term
- It has a value of 0 for only one input combination
- It is 1 for all the other combinations of variables
- That is why it is called a max (sum) term
- Each row in truth table has a max term corresponding to it
- Example, a max term \((x+y+z)\) is 0 for combination \(xyz=000\) only

### Truth Tables and Logic Expression for Adder

- \( X(A,B,C) = \sum m(1,2,4,7) \)
- \( Y(A,B,C) = \sum m(3,5,6,7) \)

### Multiple Forms and Equivalence

- Canonical Sum-of-Product form
- Canonical Product-of-sum form
- How to convert one from other?
- Minterm expansion of \( X \) to minterm expansion of \( X' \)
  - Just take the terms that are missing
  - \( X(A,B,C) = \sum m(1,2,4,7) \)
  - \( X'(A,B,C) = \sum m(3,5,6,7) \)
- Maxterm expansion of \( X \) to maxterm expansion of \( X' \)
  - Just take the terms that are missing
  - \( X(A,B,C) = \prod M( ) \)
  - \( X'(A,B,C) = \prod M( ) \)

\[ X = A'B'C + A'BC + AB'C + ABC \]
\[ Y = A'BC + AB'C + ABC' + ABC \]
Boolean Algebra

• An algebraic structure consists of
  – a set of elements {0, 1}
  – binary operators {+, .}
  – and a unary operator {'}

• Introduced by George Boole in 1854

• An effective means of describing circuits
  built with switches

  A powerful tool that can be used for
  designing and analyzing logic circuits

George Boole
1815-1864

Axioms of Boolean Algebra

1a: 0.0 = 0
1b: 1+1 = 1
2a: 1.1 = 1
2b: 0+0 = 0
3a: 0.1 = 1.0 = 0
3b: 1+0 = 0+1 = 1
4a: If x=0, then x' = 1
4b: If x=1, then x' = 0

Single-Variable Theorems

5a: x.0 = 0 Null
5b: x+x' = 1
6a: x.1 = x Identity
6b: x+x = x
7a: x.x = x Idempotency
7b: x+x' = x
8a: x.x' = 0 Complementarity
8b: x+x = x
9: \{x'\} = x Involution

Two- and Three-Variable Properties

10a: x.y = y.x Commutative
10b: x+y = y+x
11a: x.(y.z) = (x.y).z Associative
11b: x+x.y = x.y + x.z Distributive
12a: x.(y.z) = x+y . z
12b: x+y.z = (x+y).(x+z)
13a: x.x.y = x Absorption
13b: x+x-y = x
14a: x+y = x+y' DeMorgan's Theorem
14b: (x+y).(x+y') = x
15a: x+x.y = x' + y' Another form of Absorption
15b: x+y = x + y

Simplify Logic Function by Algebraic Manipulation

X Y Z F
0 0 0 0
0 0 1 1
0 1 0 1
0 1 1 1
1 0 0 0
1 0 1 1
1 1 0 1
1 1 1 1

F = X'Y'Z + X'YZ + XZ
= X'Y'Z + X'YZ + XZ by Distributive
= X'Y'Z + X'YZ + XZ by Identity
X Y Z F
F = X'Y'Z + X'YZ + XZ

X Y Z F
0 0 0 0
0 0 1 1
0 1 0 1
0 1 1 1
1 0 0 0
1 0 1 1
1 1 0 1
1 1 1 1

F = X'Y'Z + X'YZ + XZ

Principle of Duality

Dual:
– A dual of a Boolean expression is derived by replacing . by +,
  + by ., 0 by 1, and 1 by 0 and leaving variables unchanged

  In general duality: f(x1, x2, ..., xn, 0, 1, +, .) = f(x1, x2, ..., xn, 1, 0, ., +)

Principle of Duality:
– If any theorem can be proven, the dual theorem can also be
  proven.

  A meta-theorem (a theorem about theorems)

Examples:
– Multiplication and factoring:
  • (x'y)(x'z) = x.z+x.y and
  x.y+x.z = (x'y)(x'z)

– Consensus:
  • (x'y')(z',z') = x.y'z' and
DeMorgan’s Theorem in Terms of Logic Gates

\[(a) \overline{x_1 x_2} = \overline{x_1} + \overline{x_2} \]
\[(b) \overline{x_1 + x_2} = \overline{x_1} \overline{x_2} \]

Using NAND to Implement SOP

Using NOR to Implement POS

Order of Precedence of Logic Operators

- From highest precedence to lowest: NOT, AND, OR
- We can use parenthesis to change the order
- Examples:
  \[f = X' + X \cdot Y \text{ is the same as} \]
  \[f = ((X') + (X \cdot Y)) \]
  \[f = X \cdot (Y + Z) \text{ is NOT the same as} \]
  \[f = X \cdot Y + Z \]

A Typical CAD (Computer-Aided Design) System

Verilog HDL

Popular Hardware Description Languages (HDLs):
- Verilog HDL
  - More popular with US companies
  - Similar to C / Pascal programming language in syntax
- VHDL
  - More popular with European companies
  - Similar to Ada programming language in syntax
  - More “verbose” than Verilog

Uses of Verilog:
- Synthesis
- Simulation
- Verification
Verilog Syntax

- Module / Signal names:
  - Start with a letter
  - Follow by any sequence of letter, number, _ and $
  - Case sensitive
- Comment by // or /* */
- White spaces (SPACE, TAB, blank line) are ignored.

```verilog
// An example
module example1 (x1, x2, x3, f);
  input x1, x2, x3;
  output f;
  and (g, x1, x2);
  not (k, x2);
  and (h, k, x3);
  or (f, g, h);
  endmodule
```

Multiplexer Circuit

<table>
<thead>
<tr>
<th>s</th>
<th>x1</th>
<th>x2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

More compact truth-table representation

```math
f = s'.x1 + s.x2
```

Structural Specification of Logic Circuit

```verilog
module example1 (x1, x2, x3, f);
  input x1, x2, x3;
  output f;
  and (g, x1, x2);
  not (k, x2);
  and (h, k, x3);
  or (f, g, h);
endmodule
```

Another Example of Structural Specification

```verilog
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
```

Behavioral Specification Continuous Assignment

```verilog
// Structural Specification
module example1 (x1, x2, x3, f);
  input x1, x2, x3;
  output f;
  and (g, x1, x2);
  not (k, x2);
  and (h, k, x3);
  or (f, g, h);
endmodule

// Behavioral Specification
module example1 (x1, x2, x3, f);
  input x1, x2, x3;
  output f;
  assign f = (x1 & x2) | (~x2 & x3);
endmodule
```

Behavioral Specification Procedural (Sequential) Statement

```verilog
module example5 (x1, x2, x3, f);
  input x1, x2, x3;
  output f;
  reg f;
  always @(x1 or x2 or x3)
    if (x2 == 1)
      f = x1;
    else
      f = x3;
endmodule
```