Binary variables and functions

- A binary variables is a variables that take on only two discrete values 0 and 1.
- A binary logic function produces an output as an expression of its inputs. Its inputs are binary variables and/or other binary logic functions. A binary logic function evaluates to either 0 or 1 depending on the value of its inputs.

Truth tables

- A truth table is *one way* to express a logic function in a tabular form.
- Specifies the value (output) of the logic function for each possible setting of inputs $\rightarrow$ one row for each input combination.
  - **Question:** How many rows in a truth table with n inputs?
  - **Answer:** 2^n since each input can be either 0 or 1.

Truth tables

- Row of the truth table are typically arranged in an "ordered" fashion.
- Example (3-input function):

x_0	x_1	x_2	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

- Can refer to rows of the truth table as "row 0", "row 1", etc.

Logic functions

- Can also write a logic equation as an expression.
- Example... $f = \bar{x}_2\bar{x}_1\bar{x}_0 + \bar{x}_2 x_1 \bar{x}_0 + \bar{x}_2 x_1 x_0 + x_2 x_1 x_0$.
- The logic expression provides the same information as the truth table.
- To manipulate or evaluate logic expressions, we need to define some *logic operations*.

Logic operators

- Three basic logic operations: **AND**, **OR** and **NOT**.
- Basic logic operations have symbols:

Operation	**Symbol**	**Example**
AND	$\bullet$, "nothing"	$f = x_1 \bullet x_0, f = x_1 x_0$
OR	$+$	$f = x_1 + x_0$
NOT	$!, ', \neg,$ *overbar*	$f = !x, f = x', f = \neg x, f = \bar{x}$

- The behaviour of each operation is defined via a truth table.
- Operators also have precedence: parentheses (), then **NOT**, then **AND** then **OR**.
  - The purpose of parentheses is to clarify precedence.

Logical operators – **AND**

- Generates an output of 1 when *all* inputs are 1, otherwise 0.
- AND with 2-inputs...

x_0	x_1	$f = x_1 x_0$
0	0	0
0	1	0
1	0	0
1	1	1

- Generalizes to any number of inputs... AND with n-inputs...

x_0	x_1	$\cdots$	x_{n-2}	x_{n-1}	$f = x_{n-1} x_{n-2} \cdots x_1 x_0$
0	0	$\cdots$	0	0	0
0	0	$\cdots$	0	1	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
0	1	$\cdots$	1	1	0
1	0	$\cdots$	0	0	0
1	0	$\cdots$	0	1	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
1	1	$\cdots$	1	1	1

Logical operators – **OR**

- Generates an output of 1 when *any* input is 1, otherwise 0.
- OR with 2-inputs...

x_0	x_1	$f = x_1 + x_0$
0	0	0
0	1	1
1	0	1
1	1	1

- Generalizes to any number of inputs... OR with n-inputs...

x_0	x_1	$\cdots$	x_{n-2}	x_{n-1}	$f = x_{n-1} + \cdots + x_1 + x_0$
0	0	$\cdots$	0	0	0
0	0	$\cdots$	0	1	1
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
0	1	$\cdots$	1	1	1
1	0	$\cdots$	0	0	1
1	0	$\cdots$	0	1	1
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
1	1	$\cdots$	1	1	1

Logical operators – **NOT**

- Takes only a single input and "flips" (inverts, complements) the input value.
- NOT...

x	$f = !x$
0	1
1	0

Truth tables from logic expressions

- Given a logic function f, can find its truth table by evaluating f for every possible input combination.
- Example... $f = !x_1!x_0 + x_1x_0$...

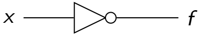
x_1	x_0	f
0	0	$f = !0 \bullet !0 + 0 \bullet 0 = 1 + 0 = 1$
0	1	$f = !0 \bullet !1 + 0 \bullet 1 = 0 + 0 = 0$
1	0	$f = !1 \bullet !0 + 1 \bullet 0 = 0 + 0 = 0$
1	1	$f = !1 \bullet !1 + 1 \bullet 1 = 0 + 1 = 1$

$\rightarrow$

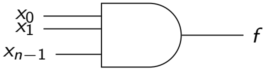
x_1	x_0	f
0	0	1
0	1	0
1	0	0
1	1	1

Schematic symbols for logic operators

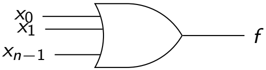
- NOT...



- AND...

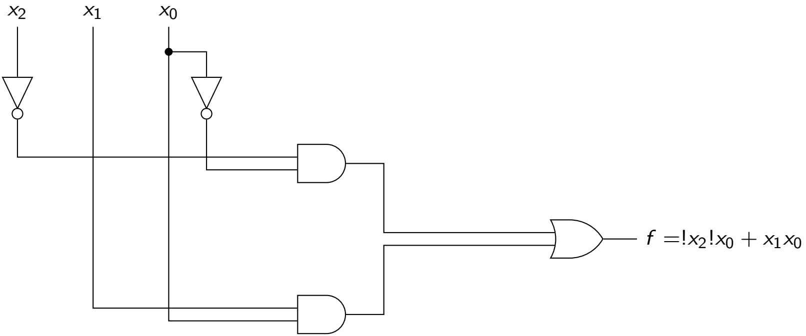


- OR...



Circuit diagrams

- Can draw diagrams for logic functions.
- Example... $f = !x_2!x_0 + x_1x_0$...



- The circuit diagram (schematic) can be seen as a *third* way to represent a logic function.
- Clearly, given a circuit diagram, we can write down the corresponding logic function.