Other types of logic gates

- Although **AND**, **OR**, and **NOT** gates are sufficient to implement any circuit, there are other useful types of logic gates.
  - "Useful" typically means that we can implement things more efficiently using these types of gates.

Logic gates — **NAND**

- **NAND** gates perform the "NOT-AND" operation (i.e., AND then NOT) — Generates an output of 0 when all inputs are 1, otherwise 0.
- NAND with 2-inputs...

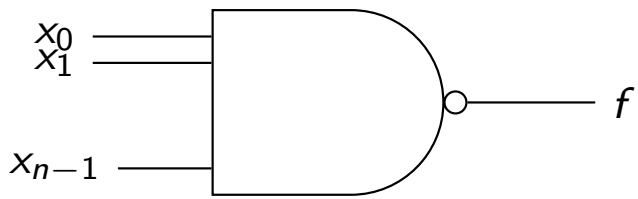
x_0	x_1	$f = \overline{x_1 x_0}$
0	0	1
0	1	1
1	0	1
1	1	0

- NAND with n-inputs...

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

- Unlike **AND**, **NAND** is NOT an associative logic gate! You cannot combine a bunch of smaller **NAND** gates into a single larger **NAND** gate.

Logic gates — **NAND**



Logical gates – **NOR**

- **NOR** gates perform the "NOT-OR " operation (i.e., OR then NOT) — Generates an output of 0 when any input is 1, otherwise 0.
- NOR with 2-inputs...

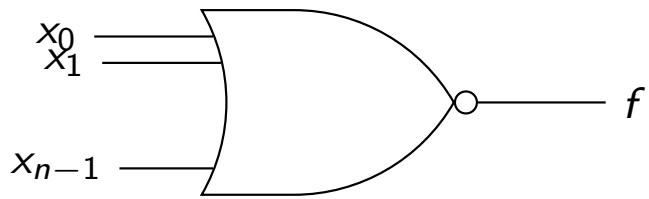
x_0	x_1	$f = \overline{x_1 + x_0}$
0	0	1
0	1	0
1	0	0
1	1	0

- NOR with n-inputs...

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

- Unlike **OR** , **NOR** is NOT an associative logic gate! You cannot combine a bunch of smaller **NOR** gates into a single larger **NOR** gate.

Logic gates — **NOR**



Logical gates – **XOR**

- **XOR** operations occur often in circuits designed for algebraic operations (e.g., adder circuits, multipler circuits...).
- XOR with 2-inputs; Generates a 1 when the inputs are *different*, otherwise 0...

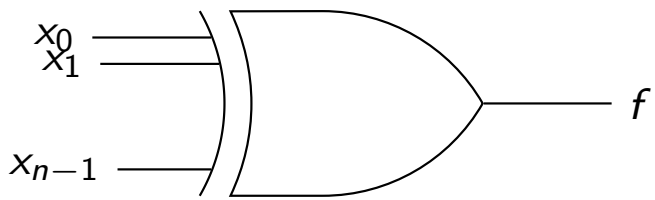
x_0	x_1	$f = x_1 \oplus x_0$
0	0	0
0	1	1
1	0	1
1	1	0

- XOR with ≥ 3 inputs; Generates a 1 when the inputs which are 1 is *odd*, otherwise 0... Example for 3-input **XOR**...

x_0	x_1	x_2	$f = x_2 \oplus x_1 \oplus x_0$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- **XOR** is associative; you can combine a bunch of **XOR** into a single **XOR** with additional inputs.

Logic gates — **XOR**



Logical gates – **NXOR**

- **NXOR** gates perform the "NOT-XOR" operation (i.e., XOR then NOT).
- **NXOR** with 2-inputs; Generates a 1 when the inputs are *equal*, otherwise 0...

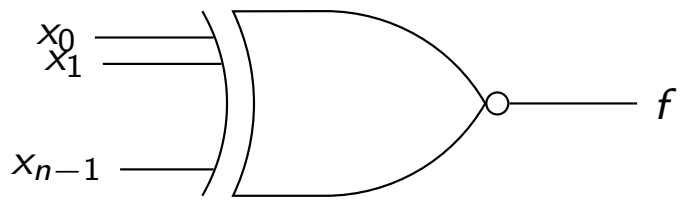
x_0	x_1	$f = \overline{x_1 \oplus x_0}$
0	0	1
0	1	0
1	0	0
1	1	1

- **NXOR** with ≥ 3 inputs; Generates a 1 when the inputs which are 1 is *even*, otherwise 0... Example for 3-input **NXOR**...

x_0	x_1	x_2	$f = \overline{x_2 \oplus x_1 \oplus x_0}$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- **NXOR** is not associative.

Logic gates — **NXOR**



Logical operators – **BUF**

- Performs a buffering operation; takes a single input and does nothing to the input logically.
- Buffers are used to boost the "driving strength" of a signal.
- BUF...

x	$f = x$
0	0
1	1