Lecture 05.03 Boolean algebra on digital signals

We will require an understanding of Boolean algebra on digital signals to implement a switch debouncing circuit in Lecture 05.04. It is a digital circuit that operates with logic gates, which are here introduced.

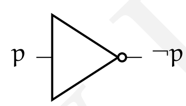
propositional calculus
Boolean algebra
truth table

A digital signal's Boolean variable values 1 and 0 are isomorphic to *propositional calculus*'s truth values $\top$ (true) and $\bot$ (false). Similarly, *Boolean algebra* (i.e. Boolean logic) operations are isomorphic to propositional calculus operations, such as *not* ($\neg$), *and* ($\wedge$), and *or* ($\vee$). Table 05.1 is a *truth table* for a number of Boolean algebra operators.

logic gates

Digital electronics instantiate these operators as *logic gates*, sometimes as subcircuits of CPUs and sometimes as discrete integrated circuits for incorporation on a prototyping board (as in Lab Exercise 05) and eventually on a PCB. The simplest gate is the *not gate*, which has the following circuit symbol.

not gate



This gate accepts digital signal represented by Boolean variable p and returns $\neg p$. So, $p = 1 \Rightarrow \neg p = 0$ and $p = 0 \Rightarrow \neg p = 1$.

or gate

Most gates have two inputs. For instance, the *or gate*, what has circuit symbol

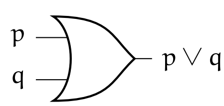
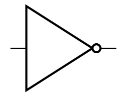
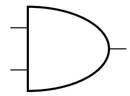
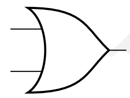
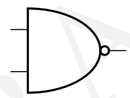
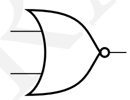
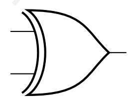
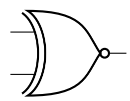


Table 05.1: a truth table for logic operations. The first two columns are operation *inputs*, the rest, *outputs*.

		not	and	or	nand	nor	xor	xnor
p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \uparrow q$	$p \downarrow q$	$p \veebar q$	$p \Leftrightarrow q$
0	0	1	0	0	1	1	0	1
0	1	1	0	1	1	0	1	0
1	0	0	0	1	1	0	1	0
1	1	0	1	1	0	0	0	1

Table 05.2: logic operations and equivalent C expressions and gate symbols.

name	logic	C	gate
not	$\neg p$	`!p`	
and	$p \wedge q$	`p&&q`	
or	$p \vee q$	`p||q`	
nand	$p \uparrow q$	`!(p&&q)`	
nor	$p \downarrow q$	`!(p||q)`	
xor	$p \veebar q$	`p!=q`	
xnor	$p \Leftrightarrow q$	`p==q`	

accepts digital signals with Boolean variables (say) p and q and returns $p \vee q$. Table 05.2 summarizes logic gates and their associated Boolean algebra operators.